ICOT Technical Memorandum: TM-0305

TM-0305

マルチPSIのベンチマーク方式

高木茂行

June, 1987

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato ku Tokyo 108 Japan

(03) 456 3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

ICOTでは並列推論マシンPIM の研究の前段階として，マルチPSI システムによる実験が行なわれつつある。マルチPSI システム及び将来のPIM の性能評価を行なうためには，一貫した思想の下での標準ベンチマーク・プログラムが必要である。並列Prologのベンチマークでは，従来の逐次型Prologのベンチマークとは異なり，要素プロセッサの機能，ネットワークの機能，総合性能，分散化アルゴリズムをそれぞれはっきり区別して評価することが必要である。本報告では，この並列Prologベンチマークの考え方及び基本方針について述べる。また並列Prologベンチマークの出発点である，Prologのベンチマークを付録に付加した。

概要

ICOTでは並列推論マシンPIM の研究の前段階として，マルチPSI システムによる実験が行なわれつつある。マルチPSI システム及び将来のPIM の性能評価を行なうためには，一貫した思想の下での標準ベンチマーク・プログラムが必要である。並列Prologのベンチマークでは，従来の逐次型Prologのベンチマークとは異なり，要素プロセッサの機能，ネットワークの機能，総合性能，分散化アルゴリズムをそれぞれはっきり区別して評価することが必要である。本報告では，この並列Prologベンチマークの考え方及び基本方針について述べる。また並列Prologベンチマークの出発点である。Prologのベンチマークを付録に付加した。

１．ベンチマークとは何か

ベンチマークと一言で言っても，それを見る者の観点によってその性格は違うと考えられる。ユーザすなわちアプリケーションの側から見れば，ベンチマークは自分が実行したい仕事のエッセンスであると言うことができる。ユーザが求めているものは，自分のプログラム全体がいかに早く終了して結果を出してくれるかであり，個々のユニフィケーションや呼出しのスピードが速いことではない。一方，ハードウェアを提供する側から見ると，ベンチマークは提供したハードウェアの性能を測定する目安である。従ってアプリケーション全体の実行性能だけではなく，ハードウェアの改良のデータとなるべき，個々の機能単位での速度や記憶領域消費量が測定できることが必要である。処理系作成者から見れば，コード生成や最適化の度合を知る目安であるから，コンパイラの持つ各種の機能を利用したプログラムであることが必要である。さらに，プログラム・コンサルタントの目で見れば，処理される問題の性質と使用されているアルゴリズムの良し悪しの目安であり，プログラムの挙動をしっかり把握できることが必要となる。

これらの目安をもとにして測定された結果から，処理系やハードウェアの性能及び機能の向上を計るとともに，プログラムに用いる言語仕様の改定，プログラムとして記述されているアルゴリズムの改訂に用いるためのデータがベンチマークである。すなわち，同一のベンチマークを用いた異なるシステムでの実行結果は，ハードウェアの相違・進歩，コンパイラの相違・進歩を示す基準となり，同じシステムにおいて，同じ問題を解く異なるプログラムによるベンチマークは，ハードウェアやコンパイラの特性，使用しているアルゴリズムの比較の基準となる。

２．Prologベンチマークの現状と分類

従来のPrologのベンチマークといえば，逐次型Prologの速度を測定，もしくは比較することが主目的であった。特にこのプロセッサ（言語プロセッサもしくはハードウェア）の能力はＸＸklips という言い方が用いられている場合，基準となっているのは，appendのスピードである。しかし，逐次型Prologのベンチマークであっても，appendのようなごく小規模のprogram ではなく，本格的なアプリケーション・プログラムによる比較が，システム全体としての本来の能力を示すものとして重要であることは，当然である。

従来の逐次型Prologのベンチマークとして，英国エジンバラ大学のWilkによる研究，NTTの奥乃によるProlog-Contest，西独ECRCのSyreらによりUSENETに公開されたベンチマーク，アメリカUCB のDespain によるBerkeley Prolog Benchmark 等がある。これらのベンチマークはProlog処理系の基本言語使用の互換性の検査，または基本機能（例えば，ユニフィケーション）の速度を求めることに重点が置かれており，並列ベンチマークに使えそうな並列度のあるプログラムは少ない。

並列Prologのベンチマークとしてよく引合いに出されるのは，n-queen とかquick sortである。明らかに，これらのプログラムは小規模で，動作特性もある程度よく把握されている。従って大規模アプリケーション・プログラムが専門でない者（コンパイラの作者やハードウェアの作者）でも扱うことができる。しかし，この程度のプログラムでは，実用的なプログラムの実行の状態を現わしているとは考えにくい。今後考えるべきベンチマークは，少なくとも数百行以上のアプリケーションを中心とすべきであろう。小規模で特性の分かったプログラムは，総合的性能の目安としては，不十分であり，個々の機能単位の測定に使われるべき物であろう。

以上のような観点から見て，ベンチマーク・プログラムは次のような分類ができる。

- 処理系とハードウェアの組合せによって提供される，プログラム言語が持つ個々の機能の処理速度を計るもの。例えばユニフィケーションの速度，述語呼出しの速度，バックトラックの速度等
- 個々の機能を用いた一定の処理を行った際の記憶領域の消費量及びそれに伴なう副次的なCPU 消費量を計るもの。例えば，GCの回数等
- 意味のある一定の処理を行うことによる総合的な性能を計るもの。すなわちあるプログラムを実行して，最終結果を得るまでのCPU 時間，記憶領域使用量等を計るもの。

これらは従来，単体の計算機と処理系の組合せによるプログラムの逐次実行を前提として，考えられてきたものである。並列処理システムのベンチマークでは，複数の要素プロセッサに関わる機能単位の性能測定，並列度の測定と並列化による性能向上（台数効果），

並列実行によるオーバーヘッド（分散化の方式に依存する性能の変化）の測定がこのほかに必要である。

３．並列Prologベンチマークの要件

並列Prologベンチマークとして必要と考えられる項目は，以下のとうりである。

- 要素プロセッサの基本処理機能の性能の測定
- プロセッサ間に渡る基本処理機能の性能の測定
- 単体プロセッサにおける総合性能の測定
- 複数プロセッサにおける総合性能の測定
- 処理の分散化率の測定（並列度）
- プロセッサ間の通信量の測定
- 分散化アルゴリズムの相違に基づく並列度，通信量の変化の測定

３．１　基本処理機能の性能の測定

個々の機能を評価するためには，評価プログラムは計りたい機能以外の余分な処理を，極力しないことが求められる。従って，ベンチマークとしては

- 単純で，
- 小さい

プログラムを何回も実行して測定することが必要である。従って個々の機能毎に，ベンチマーク・プログラムを用意することが望ましい。

ここで測定すべき項目は，

- ユニフィケーションの性能（引数の組合せ，引数の数を変えた計測）
- ゴール呼出しの性能
- バックトラックとカットの性能(Shallow, Deep)

等がある。

並列ベンチマークでは，例えば変数のユニフィケーションと言っても，要素プロセッサ内部でのユニフィケーションと，複数プロセッサを渡る通信を含めたユニフィケーションの両方を測定することが必要である。また，従来の逐次処理系ではほとんど測定されていないワーキング・セットの大きさや，プログラム／データの局所性についても測定することが必要である。特にキャッシュのヒットや仮想記憶の関係が性能に影響しないようにベンチマークを考える必要がある。

３．２　総合性能の測定

総合的性能を計るためには，現実的なすなわち実用に使用されるプログラムによる測定が必要である。必然的にベンチマークは

大規模で，
実際のアプリケーションの全部又は一部を用い，
しかもプログラムの挙動が把握できる

ことが必要である。大規模であるため，CPU や記憶領域の使用量も多く，繰返し実行による測定には向かないものが多くなる。また，キャッシュのヒット率や仮想記憶の大きさと実記憶の大きさの関係等の要素が測定結果に影響を及ぼすので，特に注意が必要となる。

大規模であるための基準として考えるべき項目の１つは，CPU 時間の消費が一定量以上等の時間的要因と，メモリ消費量が一定以上の記憶容量的要因の他，問題としての複雑さ，例えばアルゴリズムとしてのオーダ等の要因も考慮することが必要となる。PrologとAIという組合せの観点からは，自然言語処理プログラム（自然言語のパーザ等）や，定理証明系（項書換えシステム等）が適当と考えられる。

プログラムの挙動の把握は，論理モデルと実際のプログラムとの間で照合する際に是非必要であるが，大規模になるほどこれは困難になると考えられる。当面は小規模なアプリケーションから始め，次第に大規模アプリケーションによる測定へと移行するなどの手順を要すると考えられる。

総合性能として，測定すべき項目は，

プログラム終了までのCPU 時間の計測
同じく経過時間の計測（台数効果の測定）
各要素プロセッサにおける記憶領域消費量の測定（gc含む）
通信量の測定

等がある。

３．３　並列度に関する項目

並列ベンチマークでは上記の総合性能の測定のために，

並列性が存在し，

並列度が測定又は推定でき，

プログラムの挙動が規則的なものと不規則なもの

を用いて，複数のプロセッサによる実行の総合的な性能を見ることが必要である。また，台数効果と分散化アルゴリズムの関係にも注意を払うことが必要となる。

実際の並列ベンチマーク・プログラムを考える上で考慮すべき点として以下のものが考えられる。

- 用いるハードウェアと言語
- 適用する言語の対象とするアプリケーションの範囲
- プログラムの動的・静的特性の把握方法
- 処理の分散化のアルゴリズム（処理内容と分散化との分離）
- 実行測定用オーバーヘッドの測定のための空ループの作成方法
- 測定用probe の挿入による動的特性の変化
- コンパイラによる最適化の影響

そのほかに，現在のICOIでの環境条件として以下の点を考慮することが必要である。

- プログラムの持つべき並列度

  ICOIの研究では，現在は６台のMPSI，将来は100 ～1000台のPIM が評価の対象となる。従って，ベンチマークは少なくとも６台，いずれは1000台規模の並列度を実現できるものであることが必要である。即ちそれだけの計算量がある問題を選択することが必要である。もちろんプログラム自身小さくても，データによってそれなりの並列度が出るものであればかまわない。
- 用いる言語

  現在はGHC-KL1 ，将来はKL2
- AI分野におけるアプリケーション

  自然言語処理や定理証明等

これらの点を考慮したベンチマーク・プログラムを揃えていくことが課題である。

４．並列ベンチマーク作成の方針

今後並列ベンチマークを作成していく上で以下の基本方針に基づき，プログラムの作成を行なう。

- ベンチマークを
  - 要素プロセッサの処理機能単位の測定
  - プロセッサ間に渡る処理機能単位の測定
  - 総合的性能の測定
  - 分散化アルゴリズムの測定

  の４本立てとする。
- それぞれのためのベンチマーク・プログラムをできるだけ多数確保する。
- 分散化アルゴリズムの測定は当分の間行なわない。
- 総合性能測定用プログラムを用いて分散化のベンチマークを作成する。

４．１　要素プロセッサの処理機能単位の測定

要素プロセッサの処理機能単位の測定は，従来の逐次型Prologの処理機能単位の測定と変わるところは無い。そこで，従来のPrologベンチマークを利用して処理機能単位の測定を行なうこととする。内容的には，２で述べたProlog-Contestをはじめとするプログラムの中から，機能単位の測定を狙ったもののみを用いる。ただし，GHC-KL1 の場合，バックトラックとカットについては，Prologとは異なるので，それなりの工夫が必要となる。

４．２　プロセッサ間に渡る処理機能単位の測定

プロセッサ間の渡りの処理は並列マシンではじめて必要になるベンチマークである。渡りを含む個々の処理は，要素プロセッサにおいて測定された項目のそれぞれについて渡りを含む場合の測定を行なえば充分である。但し，渡りが１段の場合と複数段の場合の違いが生じることが予想されるので，１～３段程度の渡りを含む場合についてそれぞれ測定する必要があろう。また，変数のユニフィケーションについては，変数と定数がそれぞれどちら側のプロセッサにあるかによって条件が変化することが考えられるので，これについても考慮が必要である。

４．３　総合性能の測定

総合性能の測定には，ICOTにおけるアプリケーションを用いる。現在のところ，ある程度Prologによって開発が進んでいるプログラムを用いるほうが，比較の都合上具合がよいと考えられる。現在の候補としては，定理証明系のTRS，自然言語処理の BUP，SAX/PAX 等が

ある。これらのプログラムは，高速化のためにPrologのロジックでない部分(assert 等)を用いているので，それを用いない形にプログラムを書き替える必要があり，作業量がかなり多大になることが予想される。また，新規にいくつかの並列実行用の問題が提案されているので，これらの問題についても実験を行なうことが必要である。

５．問題点

実際に総合性能を測定するための並列ベンチマークを作成するにあたり，現在問題となっているのは次のような点である。

どの問題をベンチマークに用いるか
その問題を解くためのどのようなアルゴリズムを用いるか
プログラムをどうGHC で記述するか
そのプログラムの中から並列性をいかに発見するか
処理の分散化アルゴリズムをどうするか
プロセッサ間の渡りをどう指定するか
並列化が可能でも，強制的に逐次的実行を行なわせるにはどうしたら良いか

その他，並列になったがために，逐次処理ではなかった問題として，

CPU 時間や記憶領域使用の分布・共有状況の把握方法をどうするか

実測用ハードウェアが当面６台のMPSIであることによる

プログラム規模の制限に対する処理
解析的モデルとの適合方法

等がある。

５．１　問題の選択

個々の機能単位の測定は，その目的に合ったプログラムを作成することが原則であり，あまり問題は無い。問題が生じるのは，総合的な性能測定における“標準的”プログラムとは何かを決めることにある。もともと知識工学における問題は多種多用であり，特定の１個のプログラムを標準にするのは不可能と考えて良い。従って，これまでPrologで行なわれてきた各種のプログラムをGHC に変換し，その性能を測定することでこの目的に対処するのが現実的な方法である。当面は，単純なものとしてn-queen,やや複雑なものとして，

bup やpax, trs等が実験されつつあり，これらをもとに実験・計測を行なう。

５．２　解法のアルゴリズム

例えば，データのソートを行なう場合，バブル・ソートに始まって，クイック・ソート，ヒープ・ソート等，同じ問題に対する複数のアルゴリスムが知られているものがある。このような場合，どのアルゴリズムを用いるかで，実行時間は大幅に異なる場合があるほか，アルゴリズムによってはプログラミングのしやすさに影響することがしばしばある。また，並列実行の場合，むしろ難しいアルゴリズムよりも台数効果の出る単純なアルゴリズムのほうがいい場合さえもありうる。従って，同じ問題に対しても複数の解法について実験が必要となると考えられる。

５．３　ＧＨＣによるプログラミング

問題と解法としてのアルゴリズムが決まったら，それをいかにGHC で表現するかが問題となる。従来Prologで記述されてきた問題をGHC に書きかえるのは，相対的には手間は少ないが，本来の論理の部分を逸脱した機能を使用している場合や，バックトラックによる全解探索を行なっている場合には，本質的な書きなおしが必要となる事がある。

GHC ではfailが無いので，ストリームを使ったプログラミングで全解探索を行なう等の工夫が必要であり，今後ベンチマークの記述実験を通じて，この種のプログラム・スタイルを確立していくことが必要である。

５．４　並列性の発見方法

いちおうのプログラミングができあがった段階では，処理の分散化をいかに発見し，指定し，効率良い実行を行なわせるかが問題となる。プログラム・コードのどの部分を並列化すべきであるかは，個々のプログラムに依存しており，現在のところ自動的に判定する方式は確立されていない。しばらくの間この部分は人力に頼るしか方法は無いと考えられる。

５．５　分散化アルゴリズムの問題（例）

現在の６台のMPSIで並列実行を行なう場合，次のような例では，どういうアルゴリズムでゴールを外に投げるのが良いか。

```
p(X,Y):- true | q(X,R1), r(X,R2), merge(R1,R2,Y).
```

merge はR1,R2 のどちらかが確定すれば値Y を返せる場合と，両方が揃ってはじめて値を返せる場合があるものとする。q とr はp とほぼ同等の処理とすると，どのgoalを外に出すかで次のものが考えられる。

1) 外へは投げない
2) q のみ又はr のみを外に投げる
3) merge のみを外へ投げる
4) q とmerge 又はr とmerge をまとめて１台に投げる
5) q とr を別々に外へ投げ，自分はmerge のみを行なう
6) 全部を別々に外に投げる

1)はプロセッサ１台による実行であるが，スケジューリングによっては一部のゴールを省略できる可能性はある。4)と2)とは処理としては同等で，4)のほうが通信が増える。3)は1)とほぼ同等の処理である。6)は5)と同等で，通信量が増える。したがって，この場合の分散化は，2)又は5)の方法が良いと考えられる。

さらにこの問題に特殊な条件として，merge はほとんどデータ待ちの状態であり，ゴールのスケジューリングが適正に行われるとすれば，2)で充分と考えられる。

５．６　プロセッサ間の渡りの指定

現在のGHC のシンタックスでは，特定のゴールを特定のプロセッサを指定して実行させることしかできない。例えばユニィケーションのレベルでの分散化は言語プロセッサおよびハードウェアの制限から不可能である。また，分散化のアルゴリズムによって実行状況が著しく異なるという現象が既にn-queen でも観察されており，渡りの指定による実行状況の変化をいかに把握するかが今後の大きな課題である。

５．７　強制的逐次実行

並列実行と逐次実行を比較するためには，同じプログラムをただ１台のプロセッサのみを指定して実行させるだけでは不十分であり，実行を強制的に逐次的にしてしまうことが必要な場合がある。特にあるゴールによって値が決まった変数（またはストリーム）を別のゴールで使用するような場合には，タイミングの要素が大きくなるので，この機能が必須である。ところが，このような制限を行なったために記憶要領が不足して実行ができなくなる場合もあるので，プログラムの挙動を良く把握した上で計測を行なうことが必要である。プロセッサ間の渡りの影響を測定するベンチマークではこの機能が重要になると考えられる。

６．まとめ

並列推論マシンの開発のための評価用ベンチマークの方式について基本方針を述べた。並列システムの評価は従来の逐次処理システムと異なり，プロセッサ間に渡る処理の評価を含むため，各種の問題が予想される。これらを克服して総合的な評価を行なっていくことが，必要である。今後要素プロセッサの性能をはじめとする各種のベンチマークの実行・測定を行ない順次報告する予定である。

付録：逐次型Prologベンチマーク・プログラム集

ここに収録したものは，現在比較的容易に入手できるPrologのベンチマーク・プログラムである。内容は，

- ＩＣＯＴでＰＳＩの評価に用いられているプログラム
- ＮＴＴ基礎研究所（現Stanford大学）奥野によるProlog Contestのプログラム
  関連文献：情報処理学会記号処理研究会　33-4(1985.9.13)
- ＥＣＲＣのＳｙｒｅ等によるベンチマーク・プログラム
  関連文献：ＥＣＲＣ　External Report CA-24(1987.2.4)
- ＵＣＢのＤｅｓｐａｉｎ等によるBerkeley Prolog Benchmark （一部）

である。

本報告にこれらのプログラムを収録することを快諾して下さった各氏に感謝する。

照会先：

奥野　博
ＮＴＴ基礎研究所情報２研
E-mail:okuno@ntt-20.ntt.jp, okuno@sumex-aim.stanford.edu

Dr. Jean-Claude Syre
European Computer-industry Research Center, GmbH
Arabellastr. 17, D-8000
Muenchen 81, West Germany
E-mail:jclaude%ecrcvax.uucp@germany.csnet

Dr. Alvin M. Despain
503 Evans Hall, Computer Science
University of California Berkeley
CA 94720, U.S.A.
E-mail:despain@ji.berkeley.edu
Secretary:nimr@ji.berkeley.edu

ＥＣＲＣ：European Computer-industry Research Center,
a joined research center of BULL, ICL, and SIEMENS
ＵＣＢ：University of California Berkeley

From takagi Tue Apr 21 15:46:01 1987
Received: by icot.jp (1.1/6.2[Junet/CSnet])
        id AA25855; Tue, 21 Apr 87 15:45:57 JST
Date: Tue, 21 Apr 87 15:45:57 JST
From: takagi (Takagi Shigeyuki)
Message-Id: <8704210645.AA25855@icot.jp>
To: okuno@sumex-aim.stanford.edu
Subject: Prolog Contest Programs

Okuno-san:

I am writing a technical report about ICOT's parallel benchmarking guidelines. It discusses the current status of prolog benchmarks and what should be necessary fot parallel benchmarking.

Please write me a permission to include your prolog contest programs as a part of appendix of the technical report.

takagi

From takagi Tue Apr 21 15:56:45 1987
Received: by icot.jp (1.1/6.2[Junet/CSnet])
        id AA25879; Tue, 21 Apr 87 15:56:37 JST
Date: Tue, 21 Apr 87 15:56:37 JST
From: takagi (Takagi Shigeyuki)
Message-Id: <8704210656.AA25879@icot.jp>
To: michael%ecrcvax.uucp@germany.csnet
Subject: ECRC Prolog benchmark prgrams

Dear Dr. Ratcliffe,

Please forward the following to Prof. Syre.

I am writing a technical report about ICOT's parallel benchmarking guidelines. It discusses the current status of prolog benchmarks and what should be necessary for parallel becnhmarking.
(This is in Japanese for Japanese researchers. I am planning to write English version.)

I would like to add ECRC's prolog becnhmark program list
as a part of the appendix of my technical report.
Japanese researchers will be happy to have the source list for such benchmarks.

For adding ECRC programs, I need your written permission.
Please write me if you could kindly permit me to add the program list into my technical reports.

Best regards,
S.Takagi, ICOT 4th lab.

PS: According to the ARPAnet domain addressing, my E-mail address is:
        CSnet: takagi%icot.jp@relay.cs.net
        UUCP: ...!{enea,inria,ukc}!icot!takagi
        ARPA: takagi%icot.uucp@eddie.mit.edu

From takagi Tue Apr 21 16:07:23 1987
Received: by icot.jp (1.1/6.2[Junet/CSnet])
        id AA25944; Tue, 21 Apr 87 16:07:19 JST
Date: Tue, 21 Apr 87 16:07:19 JST
From: takagi (Takagi Shigeyuki)
Message-Id: <8704210707.AA25944@icot.jp>
To: touati@ernie.berkeley.edu
Subject: Berkeley Prolog benchmark

Touati-san:
Please forward the following to Prof. Despain.

I am writing a technical report about ICOT's parallel benchmark
guidelines. It discusses the current status of Prolog benchmarks
and what should be necessary for parallel benchmarking.
(This is written in Japanese for Japanese researchers. I am palnning
to write English version.)

I would like to add your benchmark program list as a part of
the appendix of my technical report.
Japanese researchers will be happy to have the source list.

For adding the program list, I need your written permission.
Please write me if you could kindly permit me to add the program list
into my technical report.

Best regards,
S.Takagi, ICOT 4th lab.

PS: According to the ARPAnet domain addressing, researchers of ICOT
(including Dr. Uchida) can be reached with the E-mail address:

CSnet: User-ID%icot.jp@relay.cs.net
UUCP: ...!mit-eddie!icot!User-ID
...!{gould9,hplabs,ihnp4,seismo}!kddlab!icot!User-ID
ARPA: User-ID%icot.uucp@eddie.mit.edu

CSnet path is most preferable, because it is most reliable and fast.

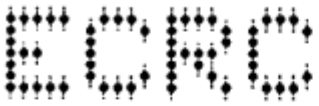
From OKUNO@sumex-aim.stanford.edu Wed Apr 22 09:18:24 1987
Received: by icot.jp (1.1/6.2[Junet/CSnet])
        id AA27816; Wed, 22 Apr 87 09:18:19 JST
Received: from CSNet-Relay by icot.jp; 22 Apr 87 9:16:10-JST (Wed)
Received: from relay.cs.net by RELAY.CS.NET id al23440; 21 Apr 87 12:26 AST
Received: from sumex-aim.stanford.edu by RELAY.CS.NET id aa06748;
21 Apr 87 12:25 EDT
Date: Tue, 21 Apr 87 09:24:19 PDT
From: Hiroshi Gitchang Okuno <Okuno@sumex-aim.stanford.edu>
Subject: Re: Prolog Contest Programs
To: takagi%icot.jp@RELAY.CS.NET
In-Reply-To: Message from "Takagi Shigeyuki <takagi%icot.jp@RELAY.CS.NET>" of Tue, 21 A
Message-Id: <12296305652.53.OKUNO@SUMEX-AIM.STANFORD.EDU>

Takagi-san,

I'm willing to give you a permission to include my Prolog Contest programs
as a part of appendix of hte technical report supposed that the reference
is given to the programs.

- Gitchang -
P.S. You may include the data of benchmarks supposed that all parts are
cited.

```
From @ji.berkeley.edu:touati%ji.Berkeley.EDU@ucbvax.berkeley.edu Wed Apr 22 09:18:43 19
Received: by icot.jp (1.1/6.2[Junet/CSnet])
        id AA27820; Wed, 22 Apr 87 09:18:37 JST
Received: from CSNet-Relay by icot.jp; 22 Apr 87 9:16:29-JST (Wed)
Received: from relay.cs.net by RELAY.CS.NET id ae23728; 21 Apr 87 13:36 AST
Received: from ji.berkeley.edu by RELAY.CS.NET id aa08044; 21 Apr 87 13:36 EDT
Received: by ji.Berkeley.EDU (5.57/1.22)
        id AA09781; Tue, 21 Apr 87 09:36:23 PST
Message-Id: <8704211736.AA09781@ji.Berkeley.EDU>
To: Takagi Shigeyuki <takagi%icot.jp@RELAY.CS.NET>
Subject: Re: Berkeley Prolog benchmark
Date: Tue, 21 Apr 87 10:36:21 PDT
From: Herve' Touati <touati%ji.Berkeley.EDU@ucbvax.berkeley.edu>

I'll transmit your request to Pr. Despain.

The list of benchmarks I sent to you is composed of small benchmarks only.
We now have a larger set of benchmarks, that are not fully tested yet.
Most of those we added are translations into Prolog of Lisp Gabriel
benchmarks. The original idea of using Gabriel benchmarks is from Evan
Tick, I believe. I got most of them from him. We did some studies
comparing the validity of our initial set of benchmarks with the
extended ones (see my paper "An Empirical Study of the WAM", to appear
in SLP '87. Well, my first paper).

I would be very interested in having your Technical Report in
Japanese. May I ask you to keep one for me? I will probably be in
Tokyo in July and August.

Best Regards,

--- Herve' Touati
PS: Do you need a written permission on paper, or email is enough?

From @ji.berkeley.edu:touati%ji.Berkeley.EDU@ucbvax.berkeley.edu Fri Apr 24 09:22:22 19
Received: by icot.jp (1.1/6.2[Junet/CSnet])
        id AA07142; Fri, 24 Apr 87 09:22:18 JST
Received: from CSNet-Relay by icot.jp; 24 Apr 87 9:16:10-JST (Fri)
Received: from relay.cs.net by RELAY.CS.NET id ag07424; 23 Apr 87 13:25 AST
Received: from ji.berkeley.edu by RELAY.CS.NET id aa17348; 23 Apr 87 13:26 EDT
Received: by ji.Berkeley.EDU (5.57/1.22)
        id AA08336; Thu, 23 Apr 87 09:25:51 PST
Message-Id: <8704231725.AA08336@ji.Berkeley.EDU>
To: Takagi Shigeyuki <takagi%icot.jp@RELAY.CS.NET>
Subject: Re: Berkeley Prolog benchmark
Date: Thu, 23 Apr 87 10:25:50 PDT
From: Herve' Touati <touati%ji.Berkeley.EDU@ucbvax.berkeley.edu>

Here is the answer of Pr Despain:

> Subject: Re:  A request from ICOT
>
> Sure, we would be pleased to have them use our benchmark programs
> and we would like to cooperate in any way we can in getting
> better benchmark programs for the whole Prolog community to use.
> Please ask if there is anything else that would help their effort
> that we could supply.
> cheers,
> al

--- Herve'
```



European Computer-Industry
Research Centre GmbH
(Forschungszentrum)

Dr. Jean-Claude Syre

Arabellastraße 17
D-8000 München 81
Germany

Tel. +49 (89) 9 26 99-127

J.-C. Syre, ECRC, Arabellastr. 17, D-8000 München 81

Dr. Takagi
ICOT 4th Research lab, Research Center
Mita-Kokusai bld. 21f
4-28 mita-1-chome, Minato-Ku
Tokyo 108
Japan

27 April 1987

Dear Dr. Takagi:

Michael Ratcliffe forwarded your message to me, I am pleased to give you the permission to include our benchmarks into your report, provided that an explicit statement indicating the origin is given (ECRC is the European Computer-industry Research Center, a joined research center of BULL, ICL and SIEMENS).

By the way, you can reach me by e-mail by substituting "michael" by "jclaude" in the e-mail address you already have for Michael Ratcliffe.

Yours sincerely,

Dr. Jean-Claude Syre

```
**********************************
*                                *
*  Benchmark Program Source List *
*                                *
*       1987.4.16  K.Nakajima    *
*                                *
**********************************
```

1.	Append 30 elements	(etimer.klC + append.klC)
2.	Append 1000 elements	(etimer.klC + append.klC)
3.	Naive Reverse	(etimer.klC + nrev.klC)
4.	Quick Sort	(etimer.klC + qsort.klC)
5.	Tree Traverse	(etimer.klC + tree.klC)
5.	LISP Interpreter (TARAI3)	(etimer.klC + lisp.klC)
7.	LISP Interpreter (FIB10)	(etimer.klC + lisp.klC)
8.	LISP Interpreter (NREVERSE)	(etimer.klC + lisp.klC)
9.	Eight Queens (1 solution)	(etimer.klC + queen.klC)
10.	Eight Queens (all solutions)	(etimer.klC + queen.klC)
11.	Reversible Function	(etimer.klC + rvsbl.klC)
12.	Slow Reverse (5 elements)	(etimer.klC + srev.klC)
13.	Slow Reverse (6 elements)	(etimer.klC + srev.klC)
14.	Harmonizer (AKATOMBO)	(etimer.klC + harmon.klC + akatom.klC)
15.	Harmonizer (MOMIJI)	(etimer.klC + harmon.klC + momiji.klC)
16.	Harmonizer (KOUJOU)	(etimer.klC + harmon.klC + koujou.klC)

From No.3 to No.13 are basically same as the benchmark program of the First Prolog Contest by Dr.Okuno.


```
% <NAKAJIMA.NSI>AKATOM.KL0.5, 14-Apr-87 00:10:22, Edit by NAKAJIMA
/*************************************************************/
/*  PSI-II Benchmark Program                                 */
/*      Harmonizer (AKATOMBO)                                */
/*              1987.1.20                                    */
/*              Post Edited by K.Nakajima                    */
/*************************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module akatom.
:- public akatom/0.
:- optimize(2).

%%%%%%%%  Top Level  %%%%%%%%

akatom :-
   display_console(a#"**** Harmonizer (AKATOMBO) ... Input Loop Count : "),
   nl,
   read_console(N),                  % set count of loop from console

   etimer:begin_watch,               % system timer clear and start
          harmon_loop(N),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_loop1(N),          % compensation loop
   etimer:end_watch(Time2),          % system timer read

   etimer:print_times(Time1,Time2,N,496,a#"AKATOMBO").
                                     % compute and print results

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                    % compensation loop
compens_loop1(_) :- dummy, fail.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).
dummy.

harmon_loop(0) :- !.
harmon_loop(_) :- akatombo, fail.
harmon_loop(N) :- N1 is N - 1, harmon_loop(N1).

%%%%%%%%  Input Harmony  %%%%%%%%

  akatombo :- gotest(1,_).

        gotest(1,O_list) :-  gen_harmony_entry(
                             1,0,1,0,0,[],
                             [
                             1,0,1,2,55,64,  %so
                             1,1,1,2,60,64,  %do
                             1,2,3,2,60,64,  %do
                             1,5,1,2,62,64,  %re
                             1,6,1,2,64,64,  %mi
                             1,7,1,2,67,64,  %so
                             1,8,1,2,72,64,  %do
                             1,9,1,2,69,64,  %ra
                             1,10,2,2,67,64, %so
                             1,12,1,2,69,64, %ra
                             1,13,1,2,60,64, %do
                             1,14,2,2,60,64, %do
                             1,16,2,2,62,64, %re
                             1,18,6,2,64,64, %mi
```

```
                    1,24,1,2,64,64, %ni
                    1,25,1,2,69,64, %ra
                    1,26,3,2,67,67, %so
                    1,29,1,2,69,64, %ra
                    1,30,1,2,72,64, %do
                    1,31,1,2,69,64, %ra
                    1,32,1,2,67,64, %so
                    1,33,1,2,69,64, %ra
                    1,34,1,2,67,64, %so
                    1,35,1,2,64,64, %mi
                    1,36,1,2,67,64, %so
                    1,37,1,2,64,64, %mi
                    1,38,1,2,60,64, %do
                    1,39,1,2,64,64, %mi
                    1,40,1,2,62,64, %re
                    1,41,1,2,60,64, %do
                    1,42,6,2,60,64, %do
                    -1 ],
                    5,O_list,L2_status);
                    !.

%%%%%%%%  Harmonizer  %%%%%%%%

        gen_harmony_entry(A,B,C,D,E,F,G,H,I,J) :- !,
                gen_harmony(A,B,C,D,E,F, -1,G,H,I,J).


gen_harmony(_, error, _, _, _, _, _, _, _, [-1], illegal_format) :- !.
gen_harmony([-1, _, _, _, _, _, _, _, _, _], [-1], normal_end] :- !.
gen_harmony(_, _, Generation_type, Key_scale, Previous_scale,
            Previous_harmony,Pre_pre_bas,I_pnt, Previous_harmony_scale,
            [1, Time, Sound_length, 8, Sop, Velocity, Alt,
            Velocity, Ten, Velocity, Bas, Velocity O_list],
            Status)
    :-  get_scale(I_pnt, Scale, End_flag, Time, Sound_length,
                                Velocity, I_pnt_pass, Error_status),
        harmon:generate_nhsent(Previous_harmony_scale, Harmony_scale,
                        Previous_scale, Scale,Key_scale),
        ending_test(End_flag, Harmony_scale,Previous_harmony_scale),
        harmon:generate(Generation_type, Key_scale,
                                Scale, Harmony_scale, Harmony, Times),

        decompose(Harmony, Sop, Alt, Ten, Bas),
        bas_ending_test(End_flag,Bas,Key_scale),
        bas_progression_test(Pre_pre_bas,Bas),
        get_pre_bas(Previous_harmony,Pre_bas),
        distribution_test(Times, Previous_harmony,
                                Harmony, Generation_type),

         harmon:progression_test(Times, Previous_harmony,
                        Harmony,Previous_harmony_scale, Harmony_scale,
                        Generation_type,Key_scale),

         gen_harmony(End_flag, Error_status, Generation_type, Key_scale,
                        Scale, Harmony, Pre_bas, I_pnt_pass,
                        Harmony_scale, O_list, Status)



get_scale([1, Time, Sound_length, 2, Scale, Velocity, End_flag|Rest],
                Scale, End_flag, Time, Sound_length, Velocity,
                [End_flag|Rest], normal) :- !.
get_scale(I_pnt, 60, 1, 0, 0, 0, I_pnt, error).
```



```
ending_test(-1,1,5) :- !.
ending_test(-1,1,4) :- !.
ending_test(-1,1,1) :- !.
ending_test(1,_,_).

bas_ending_test(-1,Bas,Key_scale) :-
        !, Key_scale =:= Bas mod 12.
bas_ending_test(_,_,_).


bas_progression_test(-1,_) :- !.
bas_progression_test(Pre_pre_bas,Bas) :-
        Bas_distance is Pre_pre_bas - Bas,
        Bas_distance > -13, Bas_distance < 13


decompose([Sop, Alt, Ten, Bas], Sop, Alt, Ten, Bas).

get_pre_bas([],-1) :- !.
get_pre_bas([_,_,_,Bas],Bas).


distribution_test(0,[ ], _, _) :- !.
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], _) :-
        Ss - Tt > 12, Sop - Ten > 12, !.
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], _) :-
        Ss - Tt < 12, Sop - Ten < 12, !.
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], Type) :-
        Ss - Tt =:= 12, !.
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], Type) :-
        Sop - Ten =:= 12, !.
distribution_test(1,_,_,_).
```

```
% APPEND.KL0.2, 10-Feb-87 18:46:19, Edit by NAKAJIMA
/*************************************************************/
/*  Benchmark Program "Append (30/1000)"                     */
/*      Append 30/1000 list elements.                        */
/*      the complexity of this computation is 30/1000 LI.    */
/*          1987.2.10    by K.Nakajima                       */
/*************************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module append.
:- public append/0.

%%%%%%%%  Top Level  %%%%%%%%

append :-  display_console(a#"***** APPEND *****"),

           display_console(a#" -------  Loop and List Type Menu -------- "),
           display_console(a#"       TRO          and   30 elem ... 0 "),
           display_console(a#"       TRO          and 1000 elem ... 1 "),
           display_console(a#"       Fail-Repeat and   30 elem ... 2 "),
           display_console(a#"       Fail-Repeat and 1000 elem ... 3 "),
           display_console(a#"***** Input Loop and List Type : "), nl,
           read_console(Ltype),      % select loop type (TRO or Fail-Repeat)

           display_console(a#"***** Input Loop Count : "), nl,
           read_console(N),          % set count of loop from console

           invoke_eval(Ltype,N).

invoke_eval(0,N) :- !, get_list30(List),
                    app_tro(N,List,30,a#"Append 30 elem (TRO)").
invoke_eval(1,N) :- !, get_list1000(List),
                    app_tro(N,List,1000,a#"Append 1000 elem (TRO)").
invoke_eval(2,N) :- !, get_list30(List),
                    app_fail(N,List,30,a#"Append 30 elem (Fail-Repeat)").
invoke_eval(3,N) :- !, get_list1000(List),
                    app_fail(N,List,1000,a#"Append 1000 elem (Fail-Repeat)").

app_tro(N,List,LI,Title) :-

   etimer:begin_watch,               % system timer clear and start
          app_tro_loop(N,List),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_tro_loop2(N,List),         % compensation loop
   etimer:end_watch(Time2),         % system timer read

   etimer:print_times(Time1,Time2,N,LI,Title).
                                     % compute and print results

app_fail(N,List,LI,Title) :-

   etimer:begin_watch,               % system timer clear and start
          app_fail_loop(N,List),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_fail_loop2(N,List),         % compensation loop
   etimer:end_watch(Time2),         % system timer read

   etimer:print_times(Time1,Time2,N,LI,Title).
                                     % compute and print results

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(2).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_tro_loop2(0,List) :- !.                    % compensation loop
compens_tro_loop2(N,List) :- N1 is N - 1, compens_tro_loop2(N1,List).

app_tro_loop(0,List) :- !.
app_tro_loop(N,List) :- append(List,[end],Result),
                        N1 is N - 1, app_tro_loop(N1,List).

compens_fail_loop2(0,List) :- !.                   % compensation loop
compens_fail_loop2(_,List) :- dummy3(List,[end],Result), fail.
compens_fail_loop2(N,List) :- N1 is N - 1, compens_fail_loop2(N1,List).
dummy3(List,_,_).

app_fail_loop(0,List) :- !.
app_fail_loop(_,List) :- append(List,[end],Result), fail.
app_fail_loop(N,List) :- N1 is N - 1, app_fail_loop(N1,List).

%%%%%%%%  List Maker  %%%%%%%%

get_list30([ 1,2,3,4,5,6,7,8,9,10,
            11,12,13,14,15,16,17,18,19,20,
            21,22,23,24,25,26,27,28,29,30]) :- !.
get_list301([201,202,203,204,205,206,207,208,209,210,
             211,212,213,214,215,216,217,218,219,220,
             221,222,223,224,225,226,227,228,229,230]) :- !.
get_list1000(List) :- !, conslist(1000,List).

get_list10001(List) :- !, conslist_long(1000,List).

conslist(0, []).
conslist(N,[1,2,3,4,5,6,7,8,9,0|L]) :- N1 is N - 10, conslist(N1,L).

conslist_long(0, []).
conslist_long(N,[100,200,300,400,500,600,700,800,900,1000|L]) :-
                                        N1 is N - 10, conslist_long(N1,L).

%%%%%%%%  Append  %%%%%%%%

append([],L,L):-!.
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

```
% <NAKAJIMA.NSI>TREE.KL0.2, 14-Apr-87 00:09:48, Edit by NAKAJIMA
/*********************************************************/
/*  Benchmark Program "Tree Traverse"                    */
/*          1987.1.19    by K.Nakajima                   */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module del.
:- public del/0.

%%%%%%%%  Top Level  %%%%%%%%

del :-
   display_console(a#"**** Tree Traverse (tree) ... Input Loop Count : "),
   nl,
   read_console(N),                    % set count of loop from console

   etimer:begin_watch,                 % system timer clear and start
          tree_loop(N),
   etimer:end_watch(Time1),            % system timer read

   etimer:begin_watch,                 % system timer clear and start
          compens_loop1(N),            % compensation loop
   etimer:end_watch(Time2),            % system timer read

   etimer:print_times(Time1,Time2,N,2122,a#"Tree Traverse").
                                       % compute and print results

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(0).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                          % compensation loop
compens_loop1(_) :- dummy, fail.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).
dummy.

tree_loop(0) :- !.
tree_loop(_) :- q111, fail.
tree_loop(N) :- N1 is N - 1, tree_loop(N1).

% [11] **** Tree traversing ****

% /* Bigapp : A non-determinate computation involving structure
%    accessing and lots of backtracking.  Written by Paul F. Wilk. */

% [11-1:] Measure the time of concat in q111.
q111 :- conslist(1000,L), !, concat(L1,L2,L), fail.

conslist(0 ,[]).
conslist(N,[1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5|L1])
     :-  less_than(0,N), N1 is N - 25, conslist(N1,L).

concat([],L,L).
concat([X|L1],L2,[X|L3]) :- concat(L1,L2,L3).
```



```
% <NAKAJIMA.NSI>ETIMER.KL0.15, 14-Apr-87 00:19:56, Edit by NAKAJIMA
/*********************************************************/
/*  Utility for Benchmark Execution Time Measurement     */
/*          1987.1.9    by K.Nakajima                    */
/*********************************************************/

:- module etimer.

:- public begin_watch/0.
:- public end_watch/1.
:- public print_times/5.

%%%%  Utility  %%%%

nl :- put_console(13), put_console(10).

display([H|T]) :- !, put_console(H), display(T).
display([]).

write(Hex) :- conv_to_dec_list(Hex,Rev_List),
              nreverse(Rev_List,List),
              display(List), nl.

conv_to_dec_list(Hex,[Digit]) :- Hex < 10, !, conv_hex(Hex,Digit).
conv_to_dec_list(Hex,[Digit|Rlist]) :-
                              divide_with_remainder(Hex,10,Quot,Rem),
                              conv_hex(Rem,Digit),
                              conv_to_dec_list(Quot,Rlist).

conv_hex(Hex,Digit) :- Digit is Hex + 16#"30".

nreverse([X|L0],L) :- !, nreverse(L0,L1), append(L1,[X],L).
nreverse([],[]):- !.

append([X|L1],L2,[X|L3]) :- !, append(L1,L2,L3).
append([],L,L):- !.

%%%%  begin_watch  %%%%

    begin_watch :- set_system_clock(0,0).

%%%%  end_watch  %%%%

    end_watch(Time) :- system_clock(_,Time).

%%%%  print_times  %%%%

    print_times(Time1,Time2,N,Unit_Li,String) :-    % print results
       TT is Time1 - Time2,
       shift_right(TT,4,Totaltime),          % timer precision is 1/16 msec
       nl,
       display_console(String), nl,                  % Title print
       put_console(a#"Total Time (msec) : "),
       write(Totaltime),                             % Total time in msec
       Time is Totaltime * 1000 / N,
       put_console(a#"Time per Loop (micro sec) : "),
       write(Time),                                  % Time per Loop in usec
       Li is N * Unit_Li * 1000,
       howmany_lips(Totaltime,Li).

    howmany_lips(0,Li) :- !,
       put_console(a#"Lips cannot be calculated because the total time is under
    howmany_lips(Totaltime,Li) :-
       Lips is Li / Totaltime,
       put_console(a#"Lips : "), write(Lips).                % Lips
```

```
% HARMON.KL0.1, 20-Jan-87 10:11:53, Edit by NAKAJIMA
/*********************************************************/
/*  PSI-II Benchmark Program                             */
/*      Harmonizer (Common Routine)                      */
/*              1987.1.20                                */
/*              Post Edited by K.Nakajima                */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module harmon.
:- public generate_nhsent/5.
:- public generate/6.
:- public progression_test/7.
:- optimize(2).

%%%%%%%%  Harmo2  %%%%%%%%

%generate_nhsent(X,X,Y,Y,_)
generate_nhsent(Previous_harmony_scale,Harmony_scale,_,Scale,Key_scale) :-
                Mod_scale is (Scale - Key_scale) mod 12,
                generate_nhs(Mod_scale,
                        Previous_harmony_scale,Harmony_scale) .


generate_nhs(0,1,4).
generate_nhs(0,1,2).            % seventh
generate_nhs(0,1,1).
generate_nhs(0,1,6) :- !.
generate_nhs(0,5,1).
generate_nhs(0,5,6) :- !.
generate_nhs(0,4,1).
generate_nhs(0,4,4).
generate_nhs(0,4,2) :- !.       % seventh
generate_nhs(0,6,4).
generate_nhs(0,6,2).
generate_nhs(0,6,6) :- !.       % seventh
generate_nhs(0,2,2) :- !.       % seventh
generate_nhs(0,3,6) :- !.

generate_nhs(2,1,5).
generate_nhs(2,1,2).
generate_nhs(2,1,3) :- !.       % seventh
generate_nhs(2,5,5) :- !.
generate_nhs(2,4,5).
generate_nhs(2,4,2) :- !.
generate_nhs(2,6,5).
generate_nhs(2,6,2) :- !.
generate_nhs(2,2,5).
generate_nhs(2,2,2).
generate_nhs(2,2,3) :- !.       % seventh

generate_nhs(4,1,3).
generate_nhs(4,1,1).
generate_nhs(4,1,6) :- !.
generate_nhs(4,5,1).
generate_nhs(4,5,6) :- !.
generate_nhs(4,4,1) :- !.
generate_nhs(4,6,1) :- !.
generate_nhs(4,2,3) :- !.
generate_nhs(4,3,6) :- !.

generate_nhs(5,1,4).
generate_nhs(5,1,2).
generate_nhs(5,1,5) :- !.       % seventh
generate_nhs(5,5,5) :- !.       % seventh
generate_nhs(5,4,2).
generate_nhs(5,4,4).
generate_nhs(5,4,5) :- !.       % seventh
generate_nhs(5,6,4).
generate_nhs(5,6,2).
generate_nhs(5,6,5) :- !.       % seventh
generate_nhs(5,2,2).
generate_nhs(5,2,5) :- !.       % seventh

generate_nhs(7,1,5).
generate_nhs(7,1,3).
generate_nhs(7,1,1).
generate_nhs(7,1,6) :- !.       % seventh
generate_nhs(7,5,1).
generate_nhs(7,5,5).
generate_nhs(7,5,6) :- !.       % seventh
generate_nhs(7,4,1).
generate_nhs(7,4,5) :- !.
generate_nhs(7,6,5).
generate_nhs(7,6,6) :- !.       % seventh
generate_nhs(7,2,5).
generate_nhs(7,2,3) :- !.
generate_nhs(7,3,6) :- !.       % seventh

generate_nhs(9,1,4).
generate_nhs(9,1,2).
generate_nhs(9,1,6) :- !.
generate_nhs(9,5,6) :- !.
generate_nhs(9,4,2).
generate_nhs(9,4,4) :- !.
generate_nhs(9,6,4).
generate_nhs(9,6,2).
generate_nhs(9,6,6) :- !.
generate_nhs(9,2,2) :- !.
generate_nhs(9,3,6) :- !.

generate_nhs(11,1,5).
generate_nhs(11,1,3) :- !.
generate_nhs(11,5,5) :- !.
generate_nhs(11,4,5) :- !.
generate_nhs(11,6,5) :- !.
generate_nhs(11,2,5).
generate_nhs(11,2,3) :- !.


generate_nhs(_,1,4).
generate_nhs(_,1,5).
generate_nhs(_,1,2).
generate_nhs(_,1,3).
generate_nhs(_,1,6).
generate_nhs(_,1,1) :- !.

generate_nhs(_,5,1).
generate_nhs(_,5,6).
generate_nhs(_,5,5) :- !.

generate_nhs(_,4,1).
generate_nhs(_,4,5).
generate_nhs(_,4,2).
generate_nhs(_,4,4) :- !.

generate_nhs(_,6,5).
generate_nhs(_,6,4).
generate_nhs(_,6,2).
generate_nhs(_,6,6) :- !.
```

```
generate_nhs(_,2,5).
generate_nhs(_,2,3).
generate_nhs(_,2,2) :- !.

generate_nhs(_,3,6) :- !.



        generate(A,B,C,D,E,0) :-
                                generate1(A,B,C,D,E)    .
        generate(A,B,C,D,E,1) :-
                                generate1(A,B,C,D,7)    .



generate1(soprano, Key_scale, Scale, Harmony_scale, Harmony)
   :- !, generate1(sopdur, Key_scale, Scale, Harmony_scale, Harmony)   .

generate1(1, Key_scale, Scale, Harmony_scale, Harmony)
   :- !, generate1(sopdur, Key_scale, Scale, Harmony_scale, Harmony)   .
% generate1(1, Key_scale, Scale, Harmony_scale, Harmony)
%    :- !,M_key_scale is Key_scale + 9,
%       generate1(sopmor, M_key_scale, Scale, Harmony_scale, Harmony).

generate1(tenor, Key_scale, Scale, Harmony_scale, Harmony)
   :- !, generate1(sopmor, Key_scale, Scale, Harmony_scale, Harmony)   .

generate1(2, Key_scale, Scale, Harmony_scale, Harmony)
   :- !, generate1(sopmor, Key_scale, Scale, Harmony_scale, Harmony)   .
% generate1(2, Key_scale, Scale, Harmony_scale, Harmony)
%    :- !,M_key_scale is Key_scale + 3,
%       generate1(sopdur, M_key_scale, Scale, Harmony_scale, Harmony).

generate1(3, Key_scale, Scale, Harmony_scale, Harmony)
   :- generate1(tendur, Key_scale, Scale, Harmony_scale, Harmony)   .
% generate1(3, Key_scale, Scale, Harmony_scale, Harmony)
%    :- !,M_key_scale is Key_scale + 9,
%       generate1(tenmor, M_key_scale, Scale, Harmony_scale, Harmony).

generate1(4, Key_scale, Scale, Harmony_scale, Harmony)
   :- generate1(tenmor, Key_scale, Scale, Harmony_scale, Harmony)   .
% generate1(4, Key_scale, Scale, Harmony_scale, Harmony)
%    :- !,M_key_scale is Key_scale + 3,
%       generate1(tendur, M_key_scale, Scale, Harmony_scale, Harmony).



generate1(sopdur,Key_scale,Scale,Harmony_scale,[Scale,Alt,Ten,Bas]) :-
        !,harmony_scale_change1(Harmony_scale,Exhs,Cgdhs),
        Indexnum is ((Scale - Key_scale - Exhs) mod 12) + Cgdhs,
        (sd_entry1(harmony_db1,Indexnum,Ss,Aa,Tt,Bb);
                sd_entry2(harmony_db2,Indexnum,Ss,Aa,Tt,Bb);
                sd_entry3(harmony_db3,Indexnum,Ss,Aa,Tt,Bb);
                sd_entry4(harmony_db4,Indexnum,Ss,Aa,Tt,Bb) ),
        Alt is Scale - Ss + Aa,
        Ten is Scale - Sa + Tt, Bas is Scale - Ss + Bb     .

generate1(sopmor,Key_scale,Scale,Harmony_scale,[Scale,Alt,Ten,Bas]) :-
        harmony_scale_change2(Harmony_scale,Exhs,Cgdhs),
        Indexnum is ((Scale - Key_scale - Exhs) mod 12) + Cgdhs,
        (sd_entry1(harmony_db1,Indexnum,Ss,Aa,Tt,Bb);
                sd_entry2(harmony_db2,Indexnum,Ss,Aa,Tt,Bb);
                sd_entry3(harmony_db3,Indexnum,Ss,Aa,Tt,Bb);
                sd_entry4(harmony_db4,Indexnum,Ss,Aa,Tt,Bb) ),
        Alt is Scale - Ss + Aa,
        Ten is Scale - Ss + Tt, Bas is Scale - Ss + Bb     .

sd_entry1(harmony_db1,Indexnum,Ss,Aa,Tt,Bb):-
        sd1(Indexnum,Ss,Aa,Tt,Bb).
sd_entry2(harmony_db2,Indexnum,Ss,Aa,Tt,Bb):-
        sd2(Indexnum,Ss,Aa,Tt,Bb).
sd_entry3(harmony_db3,Indexnum,Ss,Aa,Tt,Bb):-
        sd3(Indexnum,Ss,Aa,Tt,Bb).
sd_entry4(harmony_db4,Indexnum,Ss,Aa,Tt,Bb):-
        sd4(Indexnum,Ss,Aa,Tt,Bb).

harmony_scale_change1(1,0,0) :- !       .
harmony_scale_change1(2,2,15700)        .
harmony_scale_change1(2,2,10000) :- !   .
harmony_scale_change1(3,4,15700)        .
harmony_scale_change1(3,4,10000) :- !   .
harmony_scale_change1(4,5,0) :- !       .
harmony_scale_change1(5,7,5700) :- !    .
harmony_scale_change1(6,9,15700)        .
harmony_scale_change1(6,9,10000) :- !   .

harmony_scale_change2(1,0,10000) :- !   .
harmony_scale_change2(2,2,12700) :- !   .
harmony_scale_change2(3,3,0)            .
harmony_scale_change2(3,3,10000) :- !   .
harmony_scale_change2(4,5,10000) :- !   .
harmony_scale_change2(5,7,10000)        .
harmony_scale_change2(5,7,15700)        .
harmony_scale_change2(5,7,5700) :- !    .
harmony_scale_change2(6,8,0) :- !       .

%%%%%%%%  Harmo3  %%%%%%%%

sd1(0,0,-8,-17,-24).    %[do3,mi2,so1,do1]
sd1(0,0,-5,-20,-24).    %[do3,so2,mi1,do1]
sd1(0,0,-9,-12,-17).    %[do3,mi2,do2,so1]
sd1(0,0,-8,-17,-29).    %[do3,mi2,so1,so0]
sd1(0,0,-8,-24,-29).    %[do3,mi2,do1,so0]
sd1(0,0,-5,-20,-29).    %[do3,so2,mi1,so0]
sd1(0,0,-12,-20,-29).   %[do3,do2,mi1,so0]
sd1(0,0,-5,-8,-12).     %[do3,so2,mi2,do2]
sd1(0,0,-5,-8,-17).     %[do3,so2,mi2,so1]
sd1(0,0,-5,-12,-20).    %[do3,so2,do2,mi1]
sd1(0,0,-5,-24,-32) :- !.       %[do3,so2,do1,mi0]
sd1(4,4,-5,-12,-17).    %[mi3,so2,do2,so1]
sd1(4,4,-5,-12,-24).    %[mi3,so2,do2,do1]
sd1(4,4,-5,-17,-24).    %[mi3,so2,so1,do1]
sd1(4,4,0,-12,-17).     %[mi3,do3,do2,so1]
sd1(4,4,0,-17,-24).     %[mi3,do3,so1,do1]
sd1(4,4,0,-17,-29).     %[mi3,do3,so1,so0]
sd1(4,4,-5,-24,-29).    %[mi3,so2,do1,so0]
sd1(4,4,0,-5,-12).      %[mi3,do3,so2,do2]
sd1(4,4,0,-5,-17) :- !. %[mi3,do3,so2,so1]
sd1(7,7,0,-8,-17).      %[so3,do3,mi2,so1]
sd1(7,7,0,-8,-12).      %[so3,do3,mi2,do2]
sd1(7,7,4,-12,-17).     %[so3,mi3,do2,so1]
sd1(7,7,4,-12,-24).     %[so3,mi3,do2,do1]
sd1(7,7,4,0,-5).        %[so3,mi3,do3,so2]
sd1(7,7,4,0,-12).       %[so3,mi3,do3,do2]
sd1(7,7,4,-5,-12).      %[so3,mi3,so2,do2]
sd1(7,7,-5,-8,-12).     %[so3,so2,mi2,do2]
sd1(7,7,-5,-20,-24).    %[so3,so2,mi1,do1]
sd1(7,7,-5,-12,-20).    %[so3,so2,do2,mi1]
                                %[so3,do3,do2,mi1]
```

```
sd1(5700,7,-1,-10,-17).  %[so3,si2,re2,sol]
sd1(5700,7,-1,-13,-22).  %[so3,si2,re2,re1]
sd1(5700,7,-1,-17,-22).  %[so3,si2,sol,re1]
sd1(5700,7,-5,-13,-22).  %[so3,so2,si1,re1]
sd1(5700,7,5,-1,-10).    %[so3,fa3,si2,re2]
sd1(5700,7,5,-13,-22).   %[so3,fa3,si1,re1]
sd1(5700,7,2,-1,-5).     %[so3,re3,si2,so2]
sd1(5700,7,2,-1,-10).    %[so3,re3,si2,re2]
sd1(5700,7,2,-13,-17).   %[so3,re3,si1,sol]
sd1(5700,7,2,-13,-22).   %[so3,re3,si1,re1]
sd1(5700,7,-1,-5,-10).   %[so3,si2,so2,re2]
sd1(5700,7,2,-17,-25).   %[so3,re3,sol,si0]
sd1(5700,7,2,-5,-13).    %[so3,re3,so2,si1]
sd1(5700,7,2,-7,-13) :- !.      %[so3,re3,fa2,si1]
%sd1(5700,7,-1,-10,-19).        %[so3,si2,re2,fa1]
%sd1(5700,7,2,-13,-19).  %[so3,re3,si1,fa1]
%sd1(5700,7,2,-1,-7) :- !.      %[so3,re3,si2,fa2]
sd1(5704,11,2,-5,-17).   %[si3,re3,so2,sol]
sd1(5704,11,2,-7,-17).   %[si3,re3,fa2,sol]
sd1(5704,11,2,-10,-17).  %[si3,re3,re2,sol]
sd1(5704,11,2,-5,-10).   %[si3,re3,so2,re2]
sd1(5704,11,2,-17,-22).  %[si3,re3,sol,re1]
sd1(5704,11,7,2,-5).     %[si3,so3,re3,so2]
sd1(5704,11,7,2,-10).    %[si3,so3,re3,re2]
sd1(5704,11,7,-5,-10).   %[si3,so3,so2,re2]
sd1(5704,11,7,-10,-17).  %[si3,so3,re2,sol]
%sd1(5704,11,7,-10,-19).        %[si3,so3,re2,fa1]
%sd1(5704,11,7,2,-7).    %[si3,so3,re3,fa2]
sd1(5704,11,7,-10,-22).  %[si3,so3,re2,re1]
sd1(5704,11,5,2,-5).     %[si3,fa3,re3,so2]
sd1(5704,11,5,-5,-10).   %[si3,fa3,so2,re2]
sd1(5704,11,5,-10,-17) :- !.    %[si3,fa3,re2,sol]
sd1(5707,2,-5,-13,-17).  %[re3,so2,si1,sol]
%sd1(5707,2,-5,-13,-19).        %[re3,so2,si1,fa1]
sd1(5707,2,-5,-13,-22).  %[re3,so2,si1,re1]
sd1(5707,2,-1,-5,-10).   %[re3,si2,so2,re2]
sd1(5707,2,-1,-5,-17).   %[re3,si2,so2,sol]
sd1(5707,2,-1,-7,-17).   %[re3,si2,fa2,sol]
sd1(5707,2,-1,-10,-17).  %[re3,si2,re2,sol]
sd1(5707,2,-1,-17,-22).  %[re3,si2,sol,re1]
sd1(5707,2,-1,-17,-29).  %[re3,si2,sol,so0]
sd1(5707,2,-1,-19,-29).  %[re3,si2,fa1,so0]
sd1(5707,2,-7,-13,-17).  %[re3,fa2,si1,sol]
sd1(5707,2,-7,-25,-29).  %[re3,fa2,si0,so0]
sd1(5707,2,-10,-13,-17).        %[re3,re2,si1,sol]
sd1(5707,2,-10,-25,-29).        %[re3,re2,si0,so0]
sd1(5707,2,-7,-17,-25).  %[re3,fa2,sol,si0]
sd1(5707,2,-5,-7,-13).   %[re3,so2,fa2,si1]
sd1(5707,2,-5,-17,-25).  %[re3,so2,sol,si0]
sd1(5707,2,-5,-19,-25).  %[re3,so2,fa1,si0]
sd1(5707,2,-10,-17,-25) :- !.   %[re3,re2,sol,si0]
sd1(5710,5,-1,-10,-17).  %[fa3,si2,re2,sol]
sd1(5710,5,2,-1,-5).     %[fa3,re3,si2,so2]
sd1(5710,5,-1,-17,-22).  %[fa3,si2,sol,re1]
sd1(5710,5,-5,-13,-22).  %[fa3,so2,si1,re1]
sd1(5710,5,2,-13,-17).   %[fa3,re3,si1,sol]
sd1(5710,5,-1,-5,-10).   %[fa3,si2,so2,re2]
sd1(5710,5,2,-5,-13).    %[fa3,re3,so2,si1]
sd1(5710,5,2,-17,-25) :- !.     %[fa3,re3,sol,si0]

%%%%%%%%  Harmo4  %%%%%%%%

sd2(10000,9,0,-8,-15).   %[ra3,do3,mi2,ra1]
sd2(10000,9,0,-8,-20).   %[ra3,do3,mi2,mi1]
sd2(10000,9,0,-15,-20).  %[ra3,do3,ra1,mi1]
sd2(10000,9,-3,-12,-20).        %[ra3,ra2,do2,mi1]
sd2(10000,9,4,0,-8).     %[ra3,mi3,do3,mi2]
sd2(10000,9,4,-12,-20).  %[ra3,mi3,do2,mi1]
sd2(10000,9,0,-3,-8).    %[ra3,do3,ra2,mi2]
sd2(10000,9,-3,-8,-12).  %[ra3,ra2,mi2,do2]
sd2(10000,9,-3,-20,-24).        %[ra3,ra2,mi1,do1]
sd2(10000,9,4,-3,-12).   %[ra3,mi3,ra2,do2]
sd2(10000,9,4,-8,-12).   %[ra3,mi3,mi2,do2]
sd2(10000,9,4,-15,-24) :- !.    %[ra3,mi3,ra1,do1]
sd2(10003,0,-8,-15,-20).        %[do3,mi2,ra1,mi1]
sd2(10003,0,-8,-15,-27).        %[do3,mi2,ra1,ra0]
sd2(10003,0,-8,-20,-27).        %[do3,mi2,mi1,ra0]
sd2(10003,0,-8,-27,-32).        %[do3,mi2,ra0,mi0]
sd2(10003,0,-3,-8,-15).  %[do3,ra2,mi2,ra1]
sd2(10003,0,-3,-8,-20).  %[do3,ra2,mi2,mi1]
sd2(10003,0,-3,-15,-20).        %[do3,ra2,ra1,mi1]
sd2(10003,0,-3,-20,-27).        %[do3,ra2,mi1,ra0]
sd2(10003,0,-3,-20,-32) :- !.   %[do3,ra2,mi1,mi0]
sd2(10007,4,-3,-12,-20).        %[mi3,ra2,do2,mi1]
sd2(10007,4,0,-3,-15).   %[mi3,do3,ra2,ra1]
sd2(10007,4,0,-8,-15).   %[mi3,do3,mi2,ra1]
sd2(10007,4,0,-15,-20).  %[mi3,do3,ra1,mi1]
sd2(10007,4,0,-15,-27).  %[mi3,do3,ra1,ra0]
sd2(10007,4,0,-3,-8).    %[mi3,do3,ra2,mi2]
sd2(10007,4,-3,-15,-24).        %[mi3,ra2,ra1,do1]
sd2(10007,4,-3,-20,-24).        %[mi3,ra2,mi1,do1]
sd2(10007,4,-8,-15,-24).        %[mi3,mi2,ra1,do1]
sd2(10007,4,-3,-8,-12).  %[mi3,ra2,mi2,do2]

%%%%%%%%  Harmo5  %%%%%%%%

%sd3(15700,4,-1,-17,-22).        %[mi3,si2,sol,re1]
%sd3(15700,4,-5,-13,-22).        %[mi3,so2,si1,re1]
sd3(15700,4,-1,-10,-17).        %[mi3,si2,re2,sol]
sd3(15700,4,2,-1,-5).    %[mi3,re3,si2,so2]
sd3(15700,4,2,-5,-13).   %[mi3,re3,so2,si1]
sd3(15700,4,2,-13,-17).  %[mi3,re3,si1,sol]
sd3(15700,4,2,-17,-25) :- !.    %[mi3,re3,sol,si0]
%sd3(15700,4,-1,-5,-10) :- !.    %[mi3,si2,so2,re2]
sd3(15703,7,-1,-10,-20).        %[so3,si2,re2,mi1]
sd3(15703,7,2,-13,-20).  %[so3,re3,si1,mi1]
%sd3(15703,7,4,-1,-10).  %[so3,mi3,si2,re2]
%sd3(15703,7,4,-13,-22).        %[so3,mi3,si1,re1]
sd3(15703,7,2,-1,-8).    %[so3,re3,si2,mi2]
sd3(15703,7,2,-8,-13) :- !.     %[so3,re3,mi2,si1]
sd3(15707,11,7,2,-8).    %[si3,so3,re3,mi2]
sd3(15707,11,7,-10,-20).        %[si3,so3,re2,mi1]
sd3(15707,11,4,-10,-17).        %[si3,mi3,re2,sol]
sd3(15707,11,2,-8,-17).  %[si3,re3,mi2,sol]
sd3(15707,11,4,2,-5) :- !.      %[si3,mi3,re3,so2]
%sd3(15707,11,4,-5,-10) :- !.    %[si3,mi3,so2,re2]
sd3(15710,2,-8,-17,-25).        %[re3,mi2,sol,si0]
sd3(15710,2,-5,-8,-13).  %[re3,so2,mi2,si1]
sd3(15710,2,-5,-13,-20).        %[re3,so2,si1,mi1]
sd3(15710,2,-5,-20,-25).        %[re3,so2,mi1,si0]
sd3(15710,2,-8,-13,-17).        %[re3,mi2,si1,sol]
sd3(15710,2,-8,-25,-29).        %[re3,mi2,si0,so0]
sd3(15710,2,-1,-8,-17).  %[re3,si2,mi2,sol]
sd3(15710,2,-1,-20,-29) :- !.   %[re3,si2,mi1,so0]
sd3(12700,11,2,-7,-13).  %[si3,re3,fa2,si1]
%sd3(12700,11,2,-7,-15).        %[si3,re3,fa2,ra1]
sd3(12700,11,2,-7,-19).  %[si3,re3,fa2,fa1]
sd3(12700,11,2,-13,-19).        %[si3,re3,si1,fa1]
sd3(12700,11,2,-15,-19).        %[si3,re3,ra1,fa1]
sd3(12700,11,-1,-10,-19).       %[si3,si2,re2,fa1]
sd3(12700,11,9,2,-7).    %[si3,ra3,re3,fa2]
sd3(12700,11,9,-10,-19).        %[si3,ra3,re2,fa1]
```

```
%sd3(12700,11,5,2,-3).   %[si3,fa3,re3,ra2]
sd3(12700,11,5,2,-7).    %[si3,fa3,re3,fa2]
%sd3(12700,11,5,-10,-15).        %[si3,fa3,re2,ra1]
sd3(12700,11,5,-10,-19).         %[si3,fa3,re2,fa1]
sd3(12700,11,2,-1,-7).   %[si3,re3,si2,fa2]
sd3(12700,11,2,-3,-7).   %[si3,re3,ra2,fa2]
sd3(12700,11,5,-13,-22).         %[si3,fa3,si1,re1]
sd3(12700,11,5,-2,-10). %[si3,fa3,si2,re2]
sd3(12700,11,5,-3,-10) :- !.     %[si3,fa3,ra2,re2]
sd3(12703,2,-7,-13,-19).         %[re3,fa2,si1,fa1]
sd3(12703,2,-7,-13,-25).         %[re3,fa2,si1,si0]
sd3(12703,2,-7,-15,-25).         %[re3,fa2,ra1,si0]
sd3(12703,2,-7,-19,-25).         %[re3,fa2,fa1,si0]
sd3(12703,2,-7,-25,-31).         %[re3,fa2,si0,fa0]
sd3(12703,2,-1,-3,-7).   %[re3,si2,ra2,fa2]
sd3(12703,2,-1,-7,-13).  %[re3,si2,fa2,si1]
%sd3(12703,2,-1,-7,-15).         %[re3,si2,fa2,ra1]
sd3(12703,2,-1,-7,-19).  %[re3,si2,fa2,fa1]
sd3(12703,2,-1,-13,-19).         %[re3,si2,si1,fa1]
sd3(12703,2,-1,-15,-19).         %[re3,si2,ra1,fa1]
sd3(12703,2,-1,-19,-25).         %[re3,si2,fa1,si0]
%sd3(12703,2,-1,-19,-27).        %[re3,si2,fa1,ra0]
sd3(12703,2,-1,-19,-31).         %[re3,si2,fa1,fa0]
sd3(12703,2,-3,-7,-13).  %[re3,ra2,fa2,si1]
sd3(12703,2,-3,-13,-19).         %[re3,ra2,si1,fa1]
sd3(12703,2,-3,-19,-25) :- !.    %[re3,ra2,fa1,si0]
sd3(12706,5,-1,-10,-19).         %[fa3,si2,re2,fa1]
sd3(12706,5,2,-1,-7).    %[fa3,re3,si2,fa2]
sd3(12706,5,2,-1,-13).   %[fa3,re3,si2,si1]
sd3(12706,5,2,-3,-13).   %[fa3,re3,ra2,si1]
sd3(12706,5,2,-7,-13).   %[fa3,re3,fa2,si1]
sd3(12706,5,2,-13,-19).  %[fa3,re3,si1,fa1]
sd3(12706,5,2,-13,-25).  %[fa3,re3,si1,si0]
sd3(12706,5,2,-15,-25).  %[fa3,re3,ra1,si0]
sd3(12706,5,-1,-13,-22).         %[fa3,si2,si1,re1]
sd3(12706,5,-1,-15,-22).         %[fa3,si2,ra1,re1]
sd3(12706,5,-3,-13,-22).         %[fa3,ra2,si1,re1]
sd3(12706,5,-7,-13,-22).         %[fa3,fa2,si1,re1]
sd3(12706,5,-1,-3,-10) :- !.     %[fa3,si2,ra2,re2]
%sd3(12706,5,-1,-10,-15) :- !.   %[fa3,si2,re2,ra1]
sd3(12710,9,2,-7,-13).   %[ra3,re3,fa2,si1]
sd3(12710,9,2,-13,-19).  %[ra3,re3,si1,fa1]
sd3(12710,9,-1,-10,-19).         %[ra3,si2,re2,fa1]
sd3(12710,9,2,-1,-7).    %[ra3,re3,si2,fa2]
sd3(12710,9,5,-1,-10).   %[ra3,fa3,si2,re2]
sd3(12710,9,5,-13,-22) :- !.     %[ra3,fa3,si1,re1]

%%%%%%%%  Harmo6  %%%%%%%%

sd4(13000,0,-8,-16,-24).         %[do3,mi2,sol5,do1]
sd4(13000,0,-8,-16,-28).         %[do3,mi2,sol5,so0s]
sd4(13000,0,-8,-24,-28).         %[do3,mi2,do1,so0s]
sd4(13000,0,-12,-20,-28).        %[do3,do2,mi1,so0s]
sd4(13000,0,-4,-8,-12). %[do3,so2s,mi2,do2]
sd4(13000,0,-4,-8,-16). %[do3,so2s,mi2,sol5]
sd4(13000,0,-4,-20,-24).         %[do3,so2s,mi1,do1]
sd4(13000,0,-4,-20,-28).         %[do3,so2s,mi1,so0s]
sd4(13000,0,-8,-12,-16).         %[do3,mi2,do2,sol5]
sd4(13000,0,-12,-16,-20).        %[do3,do2,sol5,mi1]
sd4(13000,0,-12,-28,-32).        %[do3,do2,sc0s,mi0]
sd4(13000,0,-4,-12,-20).         %[do3,so2s,do2,mi1]
sd4(13000,0,-4,-16,-20) :- !.    %[do3,so2s,sol5,mi1]
sd4(13004,4,-4,-12,-16).         %[mi3,so2s,do2,sol5]
sd4(13004,4,-4,-12,-24).         %[mi3,so2s,do2,do1]
sd4(13004,4,-4,-16,-24).         %[mi3,so2s,sol5,do1]
sd4(13004,4,0,-4,-12).   %[mi3,do3,so2s,do2]
sd4(13004,4,0,-4,-16).   %[mi3,do3,so2s,sol5]
sd4(13004,4,0,-12,-16).  %[mi3,do3,do2,sol5]
sd4(13004,4,0,-16,-24).  %[mi3,do3,sol5,do1]
sd4(13004,4,0,-16,-28) :- !.     %[mi3,do3,sol5,so0s]
sd4(13008,8,0,-8,-16).   %[so3s,do3,mi2,sol5]
sd4(13008,8,0,-8,-12).   %[so3s,do3,mi2,do2]
sd4(13008,8,-4,-8,-12). %[so3s,so2s,mi2,do2]
sd4(13008,8,-4,-20,-24).         %[so3s,so2s,mi1,do1]
sd4(13008,8,4,0,-4).     %[so3s,mi3,do3,so2s]
sd4(13008,8,4,0,-12).    %[so3s,mi3,do3,do2]
sd4(13008,8,4,-4,-12).   %[so3s,mi3,so2s,do2]
sd4(13008,8,4,-12,-16). %[so3s,mi3,do2,sol5]
sd4(13008,8,4,-12,-24). %[so3s,mi3,do2,do1]
sd4(13008,8,0,-12,-20). %[so3s,do3,do2,mi1]
sd4(13008,8,0,-4,-8).    %[so3s,do3,so2s,mi2]
sd4(13008,8,0,-16,-20). %[so3s,do3,sol5,mi1]
sd4(13008,8,-4,-12,-20).         %[so3s,so2s,do2,mi1]

%%%%%%%%  Harmo7  %%%%%%%%
```

```
%-----------------------------------------------------------------------

%class progression_tester has


        progression_test(0,[],_,_,_,_,_) :- !   .
        progression_test(0,Pre_harmony,Harmony,
                        Pre_hscale,Hscale,Gen_type,Key_scale) :- !,
                sound_keeping_test(Pre_harmony,Harmony),  %text pp;30
                sop_bas_progression(Pre_harmony,Harmony),  %text pp;32
                scale_distance(Pre_harmony,Harmony),       %text pp;114
                limited_progression(Pre_harmony,Harmony,Pre_hscale,
                        Gen_type,Key_scale),       %text pp;114
                adjacent_8degree(Pre_harmony,Harmony),     %text pp;115
                adjacent_5degree(Pre_harmony,Harmony) .    %text pp;115
        progression_test(1,[],_,_,_,_,_) :- !   .
        progression_test(1,Pre_harmony,Harmony,
                        Pre_hscale,Hscale,Gen_type,Key_scale) :-
                sound_keeping_test(Pre_harmony,Harmony),   %text pp;30
                scale_distance(Pre_harmony,Harmony),       %text pp;114
                limited_progression(Pre_harmony,Harmony,Pre_hscale,
                        Gen_type,Key_scale),       %text pp;114
                adjacent_8degree(Pre_harmony,Harmony),     %text pp;115
                adjacent_5degree(Pre_harmony,Harmony) .    %text pp;115


%local


sound_keeping_test([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        Modss is Ss mod 12, Modaa is Aa mod 12,
        Modtt is Tt mod 12, Modbb is Bb mod 12,
        Modsop is Sop mod 12, Modalt is Alt mod 12,
        Modten is Ten mod 12, Modbas is Bas mod 12,
        (keeping(Modss,Modaa,Modtt,Modbb,Modsop,Modalt,Modten,Modbas),! ;
         (samekey(Modss,Modaa,Modtt,Modbb,
                Modsop,Modalt,Modten,Modbas), !, fail , true) )


samekey(X,_,_,_,_,X,_,_).
samekey(X,_,_,_,_,_,X,_).
samekey(_,X,_,_,X,_,_,_).
samekey(_,X,_,_,_,_,X,_).
samekey(_,_,X,_,_,X,_,_).
samekey(_,_,X,_,_,_,X,_).

keeping(X,_,_,_,X,_,_,_).
keeping(_,X,_,_,_,X,_,_).
keeping(_,_,X,_,_,_,X,_).


sop_bas_progression([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        samekey(Ss,Aa,Tt,Bb,Sop,Alt,Ten,Bas), !                .

sop_bas_progression([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        Ss > Sop, !, Bb < Bas                                   .

sop_bas_progression([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        Ss < Sop, Bb > Bas                                      .

.
```



```
scale_distance([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        Dss is Ss - Sop, distance_test(Dss), !, fail    .

scale_distance([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        Daa is Aa - Alt, distance_test(Daa), !, fail    .

scale_distance([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        Dtt is Tt - Ten, distance_test(Dtt), !, fail    .

scale_distance([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        Dbb is Bb - Bas, distance_test(Dbb), !, fail    .

scale_distance(_,_)                                     .


distance_test(11).                                      .
distance_test(10).
distance_test( -11 ).
distance_test( -10 ).
distance_test(X) :- X > 12.
distance_test(X) :- X < -12.




limited_progression(Pre_harmony,Harmony,5,Gen_type,Key_scale) :-
        test_seventh_note(Pre_harmony,Key_scale),
        !,
        lim_progress(Gen_type,Key_scale,Pre_harmony,Harmony)    .
limited_progression(_,_,_,_,_)                                  .

%%%%%%%%  Harmo8  %%%%%%%%

test_seventh_note([],_) :- !, fail      .
test_seventh_note([Note|Rest],Key_scale) :-
        (   (Note - Key_scale) mod 12 =:= 5, ! ;
            test_seventh_note(Rest,Key_scale)   )       .

lim_progress(_,_,[],_) :- !     .
lim_progress(1,Key_scale,[Pnote|Prest],[Note|Rest]) :- !,
        (Mod_note is (Pnote - Key_scale) mod 12, test_lim_note1(Mod_note),
         !,Distance is Pnote - Note, test_lim_progress1(Mod_note,Distance),
         lim_progress(1,Key_scale,Prest,Rest)   ;
         lim_progress(1,Key_scale,Prest,Rest)       )           .
lim_progress(2,Key_scale,[Pnote|Prest],[Note|Rest]) :- !,
        (Mod_note is (Pnote - Key_scale) mod 12, test_lim_note2(Mod_note),
         !,Distance is Pnote - Note, test_lim_progress2(Mod_note,Distance),
         lim_progress(2,Key_scale,Prest,Rest)   ;
         lim_progress(2,Key_scale,Prest,Rest)       )           .

test_lim_note1(11) :- ! .
test_lim_note1(5)       .

test_lim_note2(10) :- ! .
test_lim_note2(11) :- ! .
test_lim_note2(5)       .

test_lim_progress1(5,1) :- !    .
test_lim_progress1(5,0) :- !    .
test_lim_progress1(11,-1) :- !  .
test_lim_progress1(11,0)        .

test_lim_progress2(5,2) :- !    .
test_lim_progress2(5,0) :- !    .
test_lim_progress2(10,-2) :- !  .
```

```
test_lim_progress2(10,0) :- !    .
test_lim_progress2(11,-1) :- !   .
test_lim_progress2(11,0)         .


adjacent_8degree([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        test_8degree(Ss,Aa,Tt,Bb,Sop,Alt,Ten,Bas), !, fail
adjacent_8degree(_,_)

test_8degree(A,B,_,_,C,D,_,_) :- (A - B) mod 12 =:= 0, (C - D) mod 12 =:=0.
test_8degree(A,_,B,_,C,_,D,_) :- (A - B) mod 12 =:= 0, (C - D) mod 12 =:=0.
test_8degree(A,_,_,B,C,_,_,D) :- (A - B) mod 12 =:= 0, (C - D) mod 12 =:=0.
test_8degree(_,A,B,_,_,C,D,_) :- (A - B) mod 12 =:= 0, (C - D) mod 12 =:=0.
test_8degree(_,A,_,B,_,C,_,D) :- (A - B) mod 12 =:= 0, (C - D) mod 12 =:=0.
test_8degree(_,_,A,B,_,_,C,D) :- (A - B) mod 12 =:= 0, (C - D) mod 12 =:=0.



adjacent_5degree([Ss,Aa,Tt,Bb],[Sop,Alt,Ten,Bas]) :-
        test_5degree(Ss,Aa,Tt,Bb,Sop,Alt,Ten,Bas), !, fail
adjacent_5degree(_,_)

test_5degree(A,B,_,_,C,D,_,_) :- all_5degree(A,B), C - D =:= 7.
test_5degree(A,_,B,_,C,_,D,_) :- all_5degree(A,B), C - D =:= 7.
test_5degree(A,_,_,B,C,_,_,D) :- all_5degree(A,B), C - D =:= 7.
test_5degree(_,A,B,_,_,C,D,_) :- all_5degree(A,B), C - D =:= 7.
test_5degree(_,A,_,B,_,C,_,D) :- all_5degree(A,B), C - D =:= 7.
test_5degree(_,_,A,B,_,_,C,D) :- all_5degree(A,B), C - D =:= 7.

all_5degree(A,B) :- A - B =:= 7.
all_5degree(A,B) :- A - B =:= 6.
all_5degree(A,B) :- A - B =:= 8.
```



```
% <NAKAJIMA.NSI>KOUJOU.KL0.2, 14-Apr-87 00:10:41, Edit by NAKAJIMA
/*********************************************************/
/*  PSI-II Benchmark Program                             */
/*      Harmonizer (KOUJOUNOTSUKI)                       */
/*              1987.1.22                                */
/*              Post Edited by K.Nakajima                */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module koujou.
:- public koujou/0.
:- optimize(2).

%%%%%%%%  Top Level  %%%%%%%%

koujou :-
   display_console(a#"**** Harmonizer (KOUJOU) ... Input Loop Count : "),
   nl,
   read_console(N),                 % set count of loop from console

   etimer:begin_watch,              % system timer clear and start
         harmon_loop(N),
   etimer:end_watch(Time1),         % system timer read

   etimer:begin_watch,              % system timer clear and start
         compens_loop1(N),          % compensation loop
   etimer:end_watch(Time2),         % system timer read

   etimer:print_times(Time1,Time2,N,1000,a#"KOUJOUNOTSUKI").
                                    % compute and print results

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                      % compensation loop
compens_loop1(_) :- dummy, fail.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).
dummy.

harmon_loop(0) :- !.
harmon_loop(_) :- koujounotsuki, fail.
harmon_loop(N) :- N1 is N - 1, harmon_loop(N1).

%%%%%%%%  Input Harmony  %%%%%%%%

  koujounotsuki :- gotest(0,O_list).

        gotest(0,O_list) :-  gen_harmony_entry(
                        1,0,2,9,0,[],
                        [
                        1,0,1,2,64,64,  %mi
                        1,1,1,2,64,64,  %mi
                        1,2,1,2,69,64,  %ra
                        1,3,1,2,71,64,  %si
                        1,4,1,2,72,64,  %do
                        1,5,1,2,71,64,  %si
                        1,6,2,2,69,64,  %ra
                        1,8,1,2,65,64,  %fa
                        1,9,1,2,65,64,  %fa
                        1,10,1,2,64,64, %mi
                        1,11,1,2,62,64, %re
                        1,12,3,2,64,64, %mi
                        1,16,1,2,64,64, %mi
                        1,17,1,2,64,64, %mi
```

```
                        1,18,1,2,69,64, %ra
                        1,19,1,2,71,64, %si
                        1,20,1,2,72,64, %do
                        1,21,1,2,71,64, %si
                        1,22,2,2,69,64, %ra
                        1,24,1,2,65,64, %fa
                        1,25,1,2,62,64, %re
                        1,26,1,2,64,64, %mi
                        1,27,1,2,64,64, %mi
                        1,28,3,2,57,64, %ra
                        -1 ],
                        5.0_list,12_status),
                        !.

    gen_harmony_entry(A,B,C,D,E,F,G,H,I,J) :- !,
            gen_harmony(A,B,C,D,E,F, -1,G,H,I,J).


gen_harmony(_, error, _, _, _, _, _, _, _, [-1], illegal_format) :- !.
gen_harmony(-1, _, _, _, _, _, _, _, _, [-1], normal_end) :- !.
gen_harmony(_, _, Generation_type, Key_scale, Previous_scale,
          Previous_harmony,Pre_pre_bas,I_pnt, Previous_harmony_scale,
          [1, Time, Sound_length, 2, Sop, Velocity, Alt,
          Velocity, Ten, Velocity, Bas, Velocity|O_list],
          Status)
  :-  get_scale(I_pnt, Scale, End_flag, Time, Sound_length,
                          Velocity, I_pnt_pass, Error_status),
      harmon:generate_nhsent(Previous_harmony_scale, Harmony_scale,
                  Previous_scale, Scale,Key_scale),
      ending_test(End_flag, Harmony_scale,Previous_harmony_scale),
      harmon:generate(Generation_type, Key_scale,
                          Scale, Harmony_scale, Harmony, Times),

      decompose(Harmony, Sop, Alt, Ten, Bas),
      bas_ending_test(End_flag,Bas,Key_scale),
      bas_progression_test(Pre_pre_bas,Bas),
      get_pre_bas(Previous_harmony,Pre_bas),
      distribution_test(Times, Previous_harmony,
                          Harmony, Generation_type),

      harmon:progression_test(Times, Previous_harmony,
                  Harmony,Previous_harmony_scale, Harmony_scale,
                  Generation_type,Key_scale),

       gen_harmony(End_flag, Error_status, Generation_type, Key_scale,
                  Scale, Harmony, Pre_bas, I_pnt_pass,
                  Harmony_scale, O_list, Status).


get_scale([1, Time, Sound_length, 2, Scale, Velocity, End_flag|Rest],
          Scale, End_flag, Time, Sound_length, Velocity,
          [End_flag|Rest], normal) :- !.
get_scale(I_pnt, 60, 1, 0, 0, 0, I_pnt, error).


ending_test(-1,1,5) :- !.
ending_test(-1,1,4) :- !.
ending_test(-1,1,1) :- !.
ending_test(1,_,_).

bas_ending_test(-1,Bas,Key_scale) :-
      !, Key_scale =:= Bas mod 12.
bas_ending_test(_,_,_).
```

```
bas_progression_test(-1,_) :- !.
bas_progression_test(Pre_pre_bas,Bas) :-
        Bas_distance is Pre_pre_bas - Bas,
        Bas_distance > -13, Bas_distance < 13.


decompose([Sop, Alt, Ten, Bas], Sop, Alt, Ten, Bas).

get_pre_bas([],-1) :- !.
get_pre_bas([_,_,_,Bas],Bas).


distribution_test(0,[ ], _, _) :- !.
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], _) :-
        Ss - Tt > 12, Sop - Ten > 12, !.
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], _) :-
        Ss - Tt < 12, Sop - Ten < 12, !.
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], Type) :-
        Ss - Tt =:= 12, !.
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], Type) :-
        Sop - Ten =:= 12, !.
distribution_test(1,_,_,_).
```

```
% <NAKAJIMA.NSI>LISP.KL0.3, 14-Apr-87 00:09:20, Edit by NAKAJIMA
/*********************************************************/
/*  Benchmark Program "Lisp Interpreter"                 */
/*       "Tarai 3", "Fibonacci 10" and "Nreverse 30"     */
/*          1987.2.12    by K.Nakajima                   */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module lisp.
:- public lisp/0.

%%%%%%%%  Top Level  %%%%%%%%

lisp :-
   display_console(a#"*** Lisp Interpreter "),
   display_console(a#"*** (lisp) ... tarai3(0), fib10(1) or nrev30(2) ? : "),
   nl,
   read_console(P),          % read "program number"
   start_eval(P).


start_eval(0) :- !,
   display_console(a#"****** Tarai3 ... Loop Count : "),
   nl,
   read_console(N),                  % set count of loop from console

   etimer:begin_watch,               % system timer clear and start
          lisp_tarai_loop(N),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_loop1(N),          % compensation loop
   etimer:end_watch(Time2),          % system timer read

   etimer:print_times(Time1,Time2,N,56,a#"Lisp(Tarai3)").
                                     % compute and print results

start_eval(1) :- !,
   display_console(a#"****** Fibonacci10 ... Loop Count : "),
   nl,
   read_console(N),                  % set count of loop from console

   etimer:begin_watch,               % system timer clear and start
          lisp_fib_loop(N),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_loop1(N),          % compensation loop
   etimer:end_watch(Time2),          % system timer read

   etimer:print_times(Time1,Time2,N,7419,a#"Lisp(Fibonacci10)").
                                     % compute and print results

start_eval(2) :- !,
   display_console(a#"****** Nreverse30 ... Loop Count : "),
   nl,
   read_console(N),                  % set count of loop from console

   etimer:begin_watch,               % system timer clear and start
          lisp_nrev_loop(N),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_loop1(N),          % compensation loop
   etimer:end_watch(Time2),          % system timer read

   etimer:print_times(Time1,Time2,N,3496,a#"Lisp(Nreverse30)").
                                     % compute and print results
start_eval(_) :- !, lisp.

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(2).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                    % compensation loop
compens_loop1(_) :- dummy_0.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).

dummy_0 :- dummy(V), !, fail.
dummy(V).

lisp_tarai_loop(0) :- !.
lisp_tarai_loop(_) :- tarai3_0.
lisp_tarai_loop(N) :- N1 is N - 1, lisp_tarai_loop(N1).

tarai3_0 :- tarai3(V), !, fail.

lisp_fib_loop(0) :- !.
lisp_fib_loop(_) :- fib10_0.
lisp_fib_loop(N) :- N1 is N - 1, lisp_fib_loop(N1).

fib10_0 :- fib10(V), !, fail.

lisp_nrev_loop(0) :- !.
lisp_nrev_loop(_) :- reverse30_0.
lisp_nrev_loop(N) :- N1 is N - 1, lisp_nrev_loop(N1).

reverse30_0 :- reverse30(V), !, fail.
```

```
% [12] ******** Lisp interpreter in Prolog ********
%                    Micro Lisp in prolog

% ---- Lisp programs ----

% [12-1:] TARAI-3
tarai3(V) :-
 lisp([[label,tarai,
         [lambda,[x,y,z],
             [cond,[[greaterp,x,y],
                    [tarai,[tarai,[subl,x],y,z],
                           [tarai,[subl,y],z,x],
                           [tarai,[subl,z],x,y]]],
                   [t,y] ]],
        [quote,6],
        [quote,3],
        [quote,0] ], V).

% [12-2:] Fibonacci number for :0
fib10(V) :-
 lisp([[label,fib,
          [lambda,[n],
             [cond,[[eq,n,[quote,1]],[quote,1]],
                   [[eq,n,[quote,2]],[quote,1]],
                   [t,[plus,[fib,[subl,n]],
                            [fib,[difference,n,[quote,2]]]]]
             ]],
        [quote,10]], V).

% [12-3:] Reverse a list of 30 elements
reverse30(V) :-
 lisp([reverse,[quote,[1,2,3,4,5,6,7,8,9,0,
                       1,2,3,4,5,6,7,8,9,0,
                       1,2,3,4,5,6,7,8,9,0]]], V).
```



```
%%%%%%%%  Lisp Interpreter (micro Lisp) %%%%%%%%

apply(S,car,[[X|_]|_],X) :- !.
apply(S,cdr,[[_|Y]|_],Y) :- !.
apply(S,atom,[[_|_]|_],[]) :- !.
apply(S,atom,[X|_],t) :- atomm(X).
apply(S,cons,[X,Y|_],[X|Y]) :- !.
apply(S,eq,[X,X|_],t) :- !.
apply(S,eq,[X,Y|_],[]) :- neq(X,Y).
apply(S,null,[[]|_],t) :- !.
apply(S,subl,[X|_],Y) :- !, Y is X - 1.
apply(S,addl,[X|_],Y) :- !, Y is X + 1.
apply(S,plus,[X,Y|_],V) :- !, add(X,Y,V).
apply(S,difference,[X,Y|_],V) :- !, subtract(X,Y,V).
apply(S,times,[X,Y|_],V) :- !, multiply(X,Y,V).
apply(S,greaterp,[X,Y|_],t) :- less_than(Y,X), !.
apply(S,greaterp,[_,_|_],[]).
apply(S,zerop,[0|_],t) :- !.
apply(S,zerop,[_|_],[]).
apply(S,[lambda,X,Y|_],R,V) :- !, bind(S,X,R,S1), eval(S1,Y,V).
apply(S,[label,X,Y|_],R,V) :- !, bind(S,[X],[Y],S1), apply(S1,Y,R,V).
apply(S,A,R,V) :- value(S,A,W), apply(S,W,R,V), !.

eval(S,[quote,X|_],X) :- !.
eval(S,[cond|X],V) :- !, evcon(S,X,V).
eval(S,[X|Y],V) :- !, evlis(S,Y,W), apply(S,X,W,V).
eval(S,A,V) :- value(S,A,V), !.

evcon(S,[[X,Y|_]|_],V) :- eval(S,X,W), neq(W,[]), !, eval(S,Y,V).
evcon(S,[_|U],V) :- evcon(S,U,V), !.

evlis(S,[],[]) :- !.
evlis(S,[X Y],[U|V]) :-!, eval(S,X,U), evlis(S,Y,V).

bind(S,[],[],S) :- !.
bind(S,[A|X],[V|Y],S1) :- bind([[A|V]|S],X,Y,S1), !.

value([[A|V]|S],A,V) :- !.
value([X|S],A,V) :- value(S,A,V), !.

atomm([X|Y]) :- !, fail.
atomm(X).

neq(X,X) :- !, fail.
neq(X,Y).

lisp(X,V) :-
   eval([[],[t|t],[f],
         [reverse,[lambda,[l],[rev,l,[]]]],
         [rev,[lambda,[l,m],
                 [cond,[[atom,l],m],
                       [t,[rev,[cdr,l],
                               [cons,[car,l],m]]] ]]]
        ],X,V).
```

```
% (NAKAJIMA.NSI)MOMIJI.KL0.3, 14-Apr-87 03:10:32, Edit by NAKAJIMA
/*********************************************************/
/*  PSI-II Benchmark Program                             */
/*      Harmonizer (MOMIJI)                              */
/*              1987.1.22                                */
/*              Post Edited by K.Nakajima                */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module momiji.
:- public momiji/0.
:- optimize(2).

%%%%%%%%  Top Level  %%%%%%%%

momiji :-
   display_console(a#'**** Harmonizer (MOMIJI) ... Input Loop Count : '),
   nl,
   read_console(N),                  % set count of loop from console

   etimer:begin_watch,               % system timer clear and start
          harmon_loop(N),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_loop1(N),          % compensation loop
   etimer:end_watch(Time2),          % system timer read

   etimer:print_times(Time1,Time2,N,1000,a#"MOMIJI").
                                     % compute and print results

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                    % compensation loop
compens_loop1(_) :- dummy, fail.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).
dummy.

harmon_loop(0) :- !.
harmon_loop(_) :- do_momiji, fail.
harmon_loop(N) :- N1 is N - 1, harmon_loop(N1).

%%%%%%%%  Input Harmony  %%%%%%%%

  do_momiji :- gotest(2,O_list).

        gotest(2,O_list) :-  gen_harmony_entry(
                        1,0,1,0,0,[
                        [
                        1,0,1,2,64,64,  %mi
                        1,1,1,2,62,64,  %re
                        1,2,1,2,60,64,  %do
                        1,3,1,2,62,64,  %re
                        1,4,1,2,64,64,  %mi
                        1,5,1,2,60,64,  %do
                        1,6,1,2,55,64,  %sol
                        1,7,1,2,60,64,  %do
                        1,8,1,2,59,64,  %sil
                        1,9,1,2,60,64,  %do
                        1,10,1,2,62,64, %re
                        1,11,1,2,67,64, %so
                        1,12,1,2,64,64, %mi
                        1,13,1,2,62,64, %re
                        1,14,1,2,60,64, %do
                        1,15,1,2,62,64, %re
                        1,16,1,2,64,64, %mi
                        1,17,1,2,62,64, %re
                        1,18,1,2,60,64, %do
                        1,19,1,2,62,64, %re
                        1,20,1,2,64,64, %mi
                        1,21,1,2,60,64, %do
                        1,22,1,2,55,64, %sol
                        1,23,1,2,60,64, %do
                        1,24,1,2,59,64, %sil
                        1,25,1,2,60,64, %do
                        1,26,1,2,62,64, %re
                        1,27,1,2,67,64, %so
                        1,28,1,2,64,64, %mi
                        1,29,1,2,62,64, %re
                        1,30,1,2,60,64, %do
                        1,31,1,2,67,64, %so
                        1,32,1,2,64,64, %mi
                        1,33,1,2,65,64, %fa
                        1,34,1,2,67,64, %so
                        1,35,1,2,69,64, %ra
                        1,36,1,2,67,64, %so
                        1,37,1,2,64,64, %mi
                        1,38,1,2,67,64, %so
                        1,39,1,2,69,64, %ra
                        1,40,1,2,67,64, %so
                        1,41,1,2,64,64, %mi
                        1,42,1,2,62,64, %re
                        1,43,1,2,60,64, %do
                        1,44,1,2,62,64, %re
                        1,45,1,2,64,64, %mi
                        1,46,1,2,62,64, %re
                        1,47,1,2,67,64, %so
                        1,48,1,2,69,64, %ra
                        1,49,1,2,67,64, %so
                        1,50,1,2,64,64, %mi
                        1,51,1,2,62,64, %re
                        1,52,1,2,60,64, %do
                        1,53,1,2,55,64, %sol
                        1,54,1,2,60,64, %do
                        1,55,1,2,59,64, %sil
                        1,56,1,2,60,64, %do
                        1,57,1,2,64,64, %mi
                        1,58,1,2,62,64, %re
                        1,59,1,2,60,64, %do
                        -1 ],
                        5,O_list,L2_status),
                        !.

      gen_harmony_entry(A,B,C,D,E,F,G,H,I,J) :- !,
              gen_harmony(A,B,C,D,E,F, -1,G,H,I,J).


gen_harmony(_, error, _, _, _, _, _, _, _, [-1], illegal_format) :- ! .
gen_harmony(-1, _, _, _, _, _, _, _, _, [-1], normal_end) :- !     .
gen_harmony(_, _, Generation_type, Key_scale, Previous_scale,
            Previous_harmony,Pre_pre_bas,I_pnt, Previous_harmony_scale,
            [1, Time, Sound_length, 8, Sop, Velocity, Alt,
            Velocity, Ten, Velocity, Bas, Velocity|O_list],
            Status)
    :-   get_scale(I_pnt, Scale, End_flag, Time, Sound_length,
                                 Velocity, I_pnt_pass, Error_status),
         harmon:generate_nhsent(Previous_harmony_scale, Harmony_scale,
                        Previous_scale, Scale,Key_scale),
         ending_test(End_flag, Harmony_scale,Previous_harmony_scale),
```

```
        harmon:generate(Generation_type, Key_scale,
                        Scale, Harmony_scale, Harmony, Times),

        decompose(Harmony, Sop, Alt, Ten, Bas),
        bas_ending_test(End_flag,Bas,Key_scale),
        bas_progression_test(Pre_pre_bas,Bas),
        get_pre_bas(Previous_harmony,Pre_bas),
        distribution_test(Times, Previous_harmony,
                        Harmony, Generation_type),

        harmon:progression_test(Times, Previous_harmony,
                Harmony,Previous_harmony_scale, Harmony_scale,
                Generation_type,Key_scale),

         gen_harmony(End_flag, Error_status, Generation_type, Key_scale,
                Scale, Harmony, Pre_bas, I_pnt_pass,
                Harmony_scale, O_list, Status)



get_scale([1, Time, Sound_length, 2, Scale, Velocity, End_flag|Rest],
                Scale, End_flag, Time, Sound_length, Velocity,
                [End_flag|Rest], normal) :- !                    .
get_scale(I_pnt, 60, 1, 0, 0, 0, I_pnt, error)                   .


ending_test(-1,1,5) :- !                .
ending_test(-1,1,4) :- !                .
ending_test(-1,1,1) :- !                .
ending_test(1,_,_)      .

bas_ending_test(-1,Bas,Key_scale) :-
        !, Key_scale =:= Bas mod 12     .
bas_ending_test(_,_,_)                  .


bas_progression_test(-1,_) :- !         .
bas_progression_test(Pre_pre_bas,Bas) :-
        Bas_distance is Pre_pre_bas - Bas,
        Bas_distance > -13, Bas_distance < 13   .


decompose([Sop, Alt, Ten, Bas], Sop, Alt, Ten, Bas)              .

get_pre_bas([],-1) :- ! .
get_pre_bas([_,_,_,Bas],Bas)    .


distribution_test(0,[ ], _, _) :- !                              .
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], _) :-
        Ss - Tt > 12, Sop - Ten > 12, !                          .
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], _) :-
        Ss - Tt < 12, Sop - Ten < 12, !                          .
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], Type) :-
        Ss - Tt =:= 12, !                                        .
distribution_test(0,[Ss, Aa, Tt, Bb], [Sop, Alt, Ten, Bas], Type) :-
        Sop - Ten =:= 12, !                                      .
distribution_test(1,_,_,_)                                       .
```



```
% <NAKAJIMA.NSI>NREV.KL0.12, 14-Apr-87 00:09:58, Edit by NAKAJIMA
/*********************************************************/
/*  Benchmark Program "Naive Reverse (30)"               */
/*      Reverse 30 list elements.                        */
/*      the complexity of this computation is 496 LI.    */
/*          1987.1.8    by K.Nakajima                    */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module nrev.
:- public nrev/0.

%%%%%%%%  Top Level  %%%%%%%%

nrev :-  display_console(a#"**** NREV ... Input Loop Count : "), nl,
         read_console(N),           % set count of loop from console

   etimer:begin_watch,              % system timer clear and start
          nrev_loop(N),
   etimer:end_watch(Time1),         % system timer read

   etimer:begin_watch,              % system timer clear and start
          compens_loop1(N),         % compensation loop
   etimer:end_watch(Time2),         % system timer read

   etimer:print_times(Time1,Time2,N,496,a#"Nreverse").
                                    % compute and print results

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(2).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                      % compensation loop
compens_loop1(_) :- dummy, fail.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).
dummy.

nrev_loop(0) :- !.
nrev_loop(_) :- nreverse, fail.
nrev_loop(N) :- N1 is N - 1, nrev_loop(N1).

%%%%%%%%  Nreverse(30) ... 496 LI %%%%%%%%

nreverse :- nreverse([1,2,3,4,5,6,7,8,9,10,
                      11,12,13,14,15,16,17,18,19,20,
                      21,22,23,24,25,26,27,28,29,30],X).

nreverse([X|L0],L) :- nreverse(L0,L1), concatenate(L1,[X],L).
nreverse([],[]).

concatenate([X|L1],L2,[X|L3]) :- concatenate(L1,L2,L3).
concatenate([],L,L).
```

```
% <NAKAJIMA.NSI>QSORT.KL0.4,  8-Jan-87 11:29:07, Edit by NAKAJIMA
/*******************************************************/
/*  Benchmark Program "Quick Sort (50)"                */
/*      Sort 50 elements.                              */
/*      the complexity of this computation is 609 LI.  */
/*          1987.1.8    by K.Nakajima                  */
/*******************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module qsort.
:- public qsort/0.

%%%%%%%%  Top Level  %%%%%%%%

qsort :-  display_console(a#"**** Quick Sort ... Input Loop Count : "), nl,
          read_console(N),          % set count of loop from console
          list50(List50),           % make 50 elements List

   etimer:begin_watch,              % system timer clear and start
          qs_loop(N,List50),
   etimer:end_watch(Time1),         % system timer read

   etimer:begin_watch,              % system timer clear and start
          compens_loop2(N,_),       % compensation loop
   etimer:end_watch(Time2),         % system timer read

   etimer:print_times(Time1,Time2,N,609,a#"Qsort").
                                    % compute and print results

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(2).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop2(0,_) :- !.                    % compensation loop
compens_loop2(N,_) :- N1 is N - 1, compens_loop2(N1,_).

qs_loop(0,_) :- !.
qs_loop(N,List) :- qsort(List,Temp,[]), N1 is N - 1, qs_loop(N1,List).

list50( [27,74,17,33,94,18,46,83,65, 2,
         32,53,28,85,99,47,28,82, 6,11,
         55,29,39,81,90,37,10, 0,66,51,
          7,21,85,27,31,63,75, 4,95,99,
         11,28,61,74,18,92,40,53,59, 8] ).

%%%%%%%%  Qsort %%%%%%%%

qsort([X|L],R,R0) :- partition(L,X,L1,L2),
                     qsort(L2,R1,R0), qsort(L1,R,[X|R1]).
qsort([],R,R).

partition([X|L],Y,[X|L1],L2) :- not_less_than(Y,X), !, partition(L,Y,L1,L2).
partition([X|L],Y,L1,[X|L2]) :- partition(L,Y,L1,L2).
partition([],_,[],[]).
```



```
% <NAKAJIMA.NSI>QUEEN.KL0.5, 14-Apr-87 00:09:39, Edit by NAKAJIMA
/*******************************************************/
/*  Benchmark Program "Eight Queen"                    */
/*      Resolve 1 solution or all solutions            */
/*          1987.1.19    by K.Nakajima                 */
/*******************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module queen.
:- public queen/0.

%%%%%%%%  Top Level  %%%%%%%%

queen :-
   display_console(a#"**** Eight Queen ... 1(0) or all(1) solutions ? : "),
   nl,
   read_console(P),          % read "number of solutions"
   start_eval(P).

start_eval(0) :- !,
   display_console(a#"**** Eight Queen (One Solution) ... Loop Count : "),
   nl,
   read_console(N),                 % set count of loop from console

   etimer:begin_watch,              % system timer clear and start
          queen_one_sol_loop(N),
   etimer:end_watch(Time1),         % system timer read

   etimer:begin_watch,              % system timer clear and start
          compens_loop1(N),         % compensation loop
   etimer:end_watch(Time2),         % system timer read

   etimer:print_times(Time1,Time2,N,5750,a#"Queen(1)").
                                    % compute and print results

start_eval(1) :- !,
   display_console(a#"**** Eight Queen (All Solutions) ... Loop Count : "),
   nl,
   read_console(N),                 % set count of loop from console

   etimer:begin_watch,              % system timer clear and start
          queen_all_sols_loop(N),
   etimer:end_watch(Time1),         % system timer read

   etimer:begin_watch,              % system timer clear and start
          compens_loop1(N),         % compensation loop
   etimer:end_watch(Time2),         % system timer read

   etimer:print_times(Time1,Time2,N,100000,a#"Queen(all)").
                                    % compute and print results
start_eval(_) :- !, queen.

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(2).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                      % compensation loop
compens_loop1(_) :- dummy, fail.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).
dummy.
```

```
queen_one_sol_loop(0) :- !.
queen_one_sol_loop(_) :- ql31, fail.
queen_one_sol_loop(N) :- N1 is N - 1, queen_one_sol_loop(N1).

queen_all_sols_loop(0) :- !.
queen_all_sols_loop(_) :- ql32, fail.
queen_all_sols_loop(N) :- N1 is N - 1, queen_all_sols_loop(N1).

% [13] **** Eight Queens ****

% [13-1:] Get the first solution.
ql31 :- queen(8,L), !.

% [13-2:] Get all the solution.
ql32 :- queen(8,X), fail.

queen(N,L) :- generate(N,L1), try(N,L1,[],L,[],[]).

generate(0,[]) :- !.
generate(N,[N|L]) :- N1 is N - 1, generate(N1,L).

try(_,[],L,L,_,_) :- !.
try(M,S,L1,L,C,D) :-
    select(S,A,S1), C1 is M + A, notmem(C1,C), D1 is M - A,
    notmem(D1,D), M1 is M - 1, try(M1,S1,[A|L1],L,[C1|C],[D1|D]).

select([A|L],A,L).
select([A|L],X,[A|L1]) :- select(L,X,L1).

notmem(_,[]) :- !.
notmem(A,[B|L]) :- not_equal(A,B), notmem(A,L).
```



```
% <NAKAJIMA.NSI>RVSBL.KL0.2, 14-Apr-87 00:09:30, Edit by NAKAJIMA
/*********************************************************/
/*  Benchmark Program "Reversible Function"              */
/*          1987.2.12    by K.Nakajima                   */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module rvsbl.
:- public rvsbl/0.

%%%%%%%%  Top Level  %%%%%%%%

rvsbl :-
   display_console(a#"**** Reversible Function (rvsbl) ... Input Loop Count : ")
   nl,
   read_console(N),                  % set count of loop from console

   etimer:begin_watch,               % system timer clear and start
          rvsbl_loop(N),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_loop1(N),          % compensation loop
   etimer:end_watch(Time2),          % system timer read

   etimer:print_times(Time1,Time2,N,1301,a#"RVSBLE").
                                     % compute and print results

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(2).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                     % compensation loop
compens_loop1(_) :- dummy0.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).

dummy0 :- dummy, !, fail.
dummy.

rvsbl_loop(0) :- !.
rvsbl_loop(_) :- solve0.
rvsbl_loop(N) :- N1 is N - 1, rvsbl_loop(N1).

solve0 :- solve, !, fail.

% [15] **** Reversible functions ****

% [15-1:]
solve :- fib(X,s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(s(0)))))))))))))))),
           fact(X,Y).

sum(X,0,X).
sum(X,s(Y),s(Z)) :- sum(X,Y,Z).
prod(X,0,0).
prod(X,s(Y),Z) :- prod(X,Y,W),sum(W,X,Z).
fact(0,s(0)).
fact(s(X),Z) :- fact(X,Y),prod(s(X),Y,Z).

fib(0,s(0)).
fib(s(0),s(0)).
fib(s(s(N)),V) :- fib(s(N),X), fib(N,Y), sum(X,Y,V).
```

```
% <NAKAJIMA.NSI>SREV.KL0.9, 14-Apr-87 00:13:05, Edit by NAKAJIMA
/*********************************************************/
/*  Benchmark Program "Slow Reverse"                     */
/*          1987.2.12    by K.Nakajima                   */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module srev.
:- public srev/0.

%%%%%%%%  Top Level  %%%%%%%%

srev :-
   display_console(a#"**** Slow Reverse (srev) ****"),
   display_console(a#" Select  Srev N  ( N = 1-6 ) : "), nl,
   read_console(T),                  % select Srev n

   display_console(a#"           Input Loop Count : "),
   nl,
   read_console(N),                  % set count of loop from console

   etimer:begin_watch,               % system timer clear and start
          srev_loop(N,T),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_loop2(N,T),          % compensation loop
   etimer:end_watch(Time2),          % system timer read

   howmany_li(T,LI),                 % get the num of logical inferences

   etimer:print_times(Time1,Time2,N,LI,a#"Slow Reverse").
                                     % compute and print results

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(2).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop2(0,T) :- !.                    % compensation loop
compens_loop2(_,T) :- dummy0(T).
compens_loop2(N,T) :- N1 is N - 1, compens_loop2(N1,T).

dummy0(T) :- dummy(T), !, fail.
dummy(1) :- !.
dummy(2) :- !.
dummy(3) :- !.
dummy(4) :- !.
dummy(5) :- !.
dummy(6) :- !.

srev_loop(0,T) :- !.
srev_loop(_,T) :- sreverse(T).
srev_loop(N,T) :- N1 is N - 1, srev_loop(N1,T).

sreverse(T) :- srev(T), !, fail.

howmany_li(1,3) :- !.
howmany_li(2,13) :- !.
howmany_li(3,53) :- !.
howmany_li(4,213) :- !.
howmany_li(5,853) :- !.
```

```
howmany_li(6,3413) :- !.

% [17] **** Slow reverse ****

srev(1) :- !, srev([1],X).
srev(2) :- !, srev([1,2],X).
srev(3) :- !, srev([1,2,3],X).

srev(4) :- !, srev([1,2,3,4],X).

srev(5) :- !, srev([1,2,3,4,5],X).

srev(6) :- !, srev([1,2,3,4,5,6],X).


srev([],[]).
srev([A|X],Z) :- srev(X,Y), sapp(Y,[A],Z).

sapp([],X,X).
sapp(X,Y,Z) :- srev(X,[A|RX]),
               srev(X,Dummy),  % necessary fo fairness with Lisp
               srev(RX,RRX),
               sapp(RRX,[A|Y],Z).
```



```
% <NAKAJIMA.NSI>TREE.KL0.2, 14-Apr-87 00:09:48, Edit by NAKAJIMA
/*********************************************************/
/*  Benchmark Program "Tree Traverse"                    */
/*          1987.1.19    by K.Nakajima                   */
/*********************************************************/

%%%%%%%%  Entry  %%%%%%%%

:- module tree.
:- public tree/0.

%%%%%%%%  Top Level  %%%%%%%%

tree :-
   display_console(a#"**** Tree Traverse (tree) ... Input Loop Count : "),
   nl,
   read_console(N),                  % set count of loop from console

   etimer:begin_watch,               % system timer clear and start
          tree_loop(N),
   etimer:end_watch(Time1),          % system timer read

   etimer:begin_watch,               % system timer clear and start
          compens_loop1(N),          % compensation loop
   etimer:end_watch(Time2),          % system timer read

   etimer:print_times(Time1,Time2,N,2122,a#"Tree Traverse").
                                     % compute and print results

%%%%%%%%  Indexing Mode %%%%%%%%

:- optimize(2).

%%%%%%%%  Utility and Data  %%%%%%%%

nl :- put_console(13), put_console(10).

compens_loop1(0) :- !.                    % compensation loop
compens_loop1(_) :- dummy, fail.
compens_loop1(N) :- N1 is N - 1, compens_loop1(N1).
dummy.

tree_loop(0) :- !.
tree_loop(_) :- q111, fail.
tree_loop(N) :- N1 is N - 1, tree_loop(N1).

% [11] **** Tree traversing ****

% /* Bigapp : A non-determinate computation involving structure
%    accessing and lots of backtracking.  Written by Paul P. Wilk. */

% [11-1:] Measure the time of concat in q111.
q111 :- conslist(1000,L), !, concat(L1,L2,L), fail.

conslist(0 ,[]).
conslist(N,[1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5|L])
     :-   less_than(0,N), N1 is N - 25, conslist(N1,L).

concat([],L,L).
concat([X|L1],L2,[X|L3]) :- concat(L1,L2,L3).
```

```
% <OKUNO>BNCH1.PL.5,  7-Jul-84 12:24:11, Edit by OKUNO

% **** Prolog Instruction Level Benchmark ****

% [1] Unification of atoms

:- public q11/1, q12/1, q21/1, q22/1, q31/1, q32/1.
:- public q41/1, q42/1, q51/1, q52/1, q61/1, q62/1.
:- public q71/1, q72/1, q73/1, q74/1.

/*
To optimize the compiled code, add the next declarations:

:- mode p11(+), p12(+,+,+,+,+), p31(+), p32(+,+,+,+,+).
:- mode q11(-), q12(-), q21(-), q22(-), q31(-), q32(-).
:- mode q41(-), q42(-), q51(-), q52(-), q61(-), q62(-).
:- mode q71(-), q72(-), q73(-), q74(-).
:- fastcode.
:- compactcode.
*/

p11(a).
p12(a,a,a,a,a).

/*
[1-1:] Arity of one
       do "q11(1000)." for one thousand iterations.
*/

q11(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q11(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '),write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_dummy(N,N) :- !.
loop_dummy(I,N) :-
     I1 is I+1, !, loop_dummy(I1,N).

loop_q11(N,N) :- !.
loop_q11(I,N) :-
     I1 is I+1, p11(a), !, loop_q11(I1,N).

/*
[1-2:] Arity of five
       do "q12(1000)." for one thousand iterations.
*/

q12(N) :-
     statistics(garbage_collection,[_,_ G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q12(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q12(N,N) :- !.
loop_q12(I,N) :-
     I1 is I+1, p12(a,a,a,a,a), !, loop_q12(I1,N).

% [2] Unification of variables

p21(X).
p22(X,X,X,X,X).

/*
[2-1:] Arity of one
       do "q21(1000)." for one thousand iterations.
*/

q21(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q21(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q21(N,N) :- !.
loop_q21(I,N) :-
     I1 is I+1, p21(X), !, loop_q21(I1,N).

/*
[2-2:] Arity of five
       do "q22(1000)." for one thousand iterations.
*/

q22(N) :-
     statistics(garbage_collection,[_,_ G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q22(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
```

```
        write('Total = '), write(Total),
        write('ms,  runtime = '), write(T3),
        write('ms,  gctime =  '), write(G4),
        write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q22(N,N) :- !.
loop_q22(I,N) :-
        I1 is I+1, p22(X,X,X,X,X), !, loop_q22(I1,N).

% [3] Unification of constant structure

p31(f(a)).
p32(f(a),f(a),f(a),f(a),f(a)).

/*
[3-1:] Arity of one
       do "q31(1000)." for one thousand iterations.
*/

q31(N) :-
        statistics(garbage_collection,[_,_|G1]),!,
        statistics(runtime,[_,_]),!,
        loop_q31(0,N),
        statistics(runtime,[_,T1]),!,
        statistics(garbage_collection,[_,_|G2]),!,
        statistics(runtime,[_,_]),!,
        loop_dummy(0,N),
        statistics(runtime,[_,T2]),
        statistics(garbage_collection,[_,_|G3]),!,
        G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
        G4 is Gt2 + Gt2 - Gt1 - Gt3,
        T3 is T1-T2-G4, Total is T1-T2,
        write('Total = '), write(Total),
        write('ms,  runtime = '), write(T3),
        write('ms,  gctime = '), write(G4),
        write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q31(N,N) :- !.
loop_q31(I,N) :-
        I1 is I+1, p31(f(a)), !, loop_q31(I1,N).

/*
[3-2:] Arity of five
       do "q32(1000)." for one thousand iterations.
*/

q32(N) :-
        statistics(garbage_collection,[_,_|G1]),!,
        statistics(runtime,[_,_]),!,
        loop_q32(0,N),
        statistics(runtime,[_,T1]),!,
        statistics(garbage_collection,[_,_|G2]),!,
        statistics(runtime,[_,_]),!,
        loop_dummy(0,N),
        statistics(runtime,[_,T2]),
        statistics(garbage_collection,[_,_|G3]),!,
        G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
        G4 is Gt2 + Gt2 - Gt1 - Gt3,
        T3 is T1-T2-G4, Total is T1-T2,
        write('Total = '), write(Total),
        write('ms,  runtime = '), write(T3),
        write('ms,  gctime = '), write(G4),
        write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q32(N,N) :- !.
loop_q32(I,N) :-
        I1 is I+1, p32(f(a),f(a),f(a),f(a),f(a)), !, loop_q32(I1,N).

% [4] Unification of structures with variables

p41(f(X)).
p42(f(X),f(X),f(X),f(X),f(X)).

/*
[4-1:] Arity of one
       do "q41(1000)." for one thousand iterations.
*/

q41(N) :-
        statistics(garbage_collection,[_,_|G1]),!,
        statistics(runtime,[_,_]),!,
        loop_q41(0,N),
        statistics(runtime,[_,T1]),!,
        statistics(garbage_collection,[_,_|G2]),!,
        statistics(runtime,[_,_]),!,
        loop_dummy(0,N),
        statistics(runtime,[_,T2]),
        statistics(garbage_collection,[_,_|G3]),!,
        G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
        G4 is Gt2 + Gt2 - Gt1 - Gt3,
        T3 is T1-T2-G4, Total is T1-T2,
        write('Total = '), write(Total),
        write('ms,  runtime = '), write(T3),
        write('ms,  gctime = '), write(G4),
        write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q41(N,N) :- !.
loop_q41(I,N) :-
        I1 is I+1, p41(f(X)), !, loop_q41(I1,N).

/*
[4-2:] Arity of five
       do "q42(1000)." for one thousand iterations.
*/

q42(N) :-
        statistics(garbage_collection,[_,_|G1]),!,
        statistics(runtime,[_,_]),!,
        loop_q42(0,N),
        statistics(runtime,[_,T1]),!,
        statistics(garbage_collection,[_,_|G2]),!,
        statistics(runtime,[_,_]),!,
        loop_dummy(0,N),
        statistics(runtime,[_,T2]),
        statistics(garbage_collection,[_,_|G3]),!,
        G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
        G4 is Gt2 + Gt2 - Gt1 - Gt3,
        T3 is T1-T2-G4, Total is T1-T2,
        write('Total = '), write(Total),
        write('ms,  runtime = '), write(T3),
        write('ms,  gctime = '), write(G4),
        write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q42(N,N) :- !.
loop_q42(I,N) :-
        I1 is I+1, p42(f(X),f(X),f(X),f(X),f(X)), !, loop_q42(I1,N).

% [5] Unification of variables with structure

p51(f(X)).
p52(f(X),f(X),f(X),f(X),f(X)).
```

```
/*
[5-1:] Arity of one
       do "q51(1000)." for one thousand iterations.
*/

q51(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q51(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total =  '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q51(N,N) :- !.
loop_q51(I,N) :-
     I1 is I+1, p51(X), !, loop_q51(I1,N).

/*
[5-2:] Arity of five
       do "q52(1000)." for one thousand iterations.
*/

q52(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q52(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q52(N,N) :- !.
loop_q52(I,N) :-
     I1 is I+1, p52(X,X,X,X,X), !, loop_q52(I1,N).

% [6]

p61(X).
p62(X,X,X,X,X).

/*
[6-1:] Arity of one
       do "q61(1000)." for one thousand iterations.
*/

q61(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q61(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q61(N,N) :- !.
loop_q61(I,N) :-
     I1 is I+1, p61(f(X)), !, loop_q61(I1,N).

/*
[6-2:] Arity of five
       do "q62(1000)." for one thousand iterations.
*/

q62(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q62(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q62(N,N) :- !.
loop_q62(I,N) :-
     I1 is I+1, p62(f(X),f(X),f(X),f(X),f(X)), !, loop_q62(I1,N).

% [7] **** Clause Call/Return ****

p71(X) :- f71(X).
f71(X).

p72(X) :- f72(X).
f72(X) :- fail.
p72(X).

p73(X) :- fail.
p73(X) :- fail.
p73(X) :- fail.
p73(X) :- fail.
p73(X) :- fail.
p73(X) :- fail.
p73(X) :- fail.
p73(X) :- fail.
p73(X) :- fail.
p73(X) :- fail.
```

```
p73(X).

p74(X) :- f74(X),r74(X).
p74(X) :- a74(X),b74(X).
f74(X) :- s74(X),r74(X).
f74(X) :- s74(X),r74(X).
s74(X) :- r74(X).
s74(X) :- r74(X).
r74(X) :- fail.
r74(X) :- fail.
a74(X) :- s74(X),r74(X).
a74(X) :- b74(X),b74(X).
b74(X) :- fail.
b74(X).

/*
[7-1:] Deterministic simple call
       do "q71(1000)." for one thousand iterations.
*/

q71(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q71(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q71(N,N) :- !.
loop_q71(I,N) :-
     I1 is I+1, p71(X), !, loop_q71(I1,N).

/*
% [7-2:] Nondeterministic simple call
       do "q72(1000)." for one thousand iterations.
*/

q72(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q72(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q72(N,N) :- !.
loop_q72(I,N) :-
     I1 is I+1, p72(X), !, loop_q72(I1,N).

/*
[7-3] Sallow backtracking
       do "q73(1000)." for one thousand iterations.
*/

q73(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q73(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is I1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q73(N,N) :- !.
loop_q73(I,N) :-
     I1 is I+1, p73(X), !, loop_q73(I1,N).

/*
[7-4] Deep backtracking
       do "q74(1000)." for one thousand iterations.
*/

q74(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q74(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q74(N,N) :- !.
loop_q74(I,N) :-
     I1 is I+1, p74(X), !, loop_q74(I1,N).


/* Now measure the benchmarks.
   Replace 1000 by 10000 for compiled codes.
*/

q11(1000).
q12(1000).
q21(1000).
q22(1000).
```

```
q31(1000).
q32(1000).
q41(1000).
q42(1000).
q51(1000).
q52(1000).
q61(1000).
q62(1000).
q71(1000).
q72(1000).
q73(1000).
q74(1000).
```



```
% <OKUNO>BNCH2.PL.2,  7-Jul-84 11:36:03, Edit by OKUNO

% [8] **** Clause Indexing ****
/*
The following clauses are part of places database.
   country(moscow,ussr). ===>220 clauses.
*/

:- public country/2.
:- public q81/1, q82/1, q83/1, q84/1, q85/1, q86/1.

/*
To optimize the compiled code, add the next declarations:

:- mode q81(-), q82(-), q83(-), q84(-), q85(-), q86(-).
:- fastcode.
:- compactcode.
*/

country(moscow,ussr).
country(novaya_zemlya,ussr).
country(sverdlovsk,ussr).
country(vladivostok,ussr).
country(gorki,ussr).
country(novosibirsk,ussr).
country(syracuse,usa).
country(new_york_city,usa).
country(ithaca,usa).
country(albany,usa).
country(san_francisco,usa).
country(san_diego,usa).
country(washington,usa).
country(boston,usa).
country(rome_ny,usa).
country(rome,italy).
country(paris,france).
country(london,uk).
country(dublin,ireland).
country(stockholm,sweden).
country(copenhagen,denmark).
country(amsterdam,netherlands).
country(brussels,belgium).
country(madrid,spain).
country(athens,greece).
country(ankara,turkey).
country(istanbul,turkey).
country(tirane,albania).
country(sofia,bulgaria).
country(belgrade,yugoslavia).
country(warsaw,poland).
country(prague,czechoslovakia).
country(lisbon,portugal).
country(tehran,iran).
country(delhi,india).
country(islamabad,pakistan).
country(tokyo,japan).
country(brisbane,australia).
country(canberra,australia).
country(wellington,new_zealand).
country(djakarta,indonesia).
country(singapore,singapore).
country(peking,china).
country(hanoi,vietnam).
country(seoul,south_korea).
country(pyong_yang,north_korea).
country(recife,brazil).
```

```
country(brasilia,brazil).
country(santiago,chile).
country(oslo,norway).
country(vancouver,canada).
country(ottawa,canada).
country(montreal,canada).
country(toronto,canada).
country(havana,cuba).
country(rio_de_janeiro,brazil).
country(san_paolo,brazil).
country(buenos_aires,argentina).
country(tierra_del_fuego,argentina).
country(punta_arenas,chile).
country(caracas,venezuela).
country(san_juan,usa).
country(tampa,usa).
country(rangoon,burma).
country(bonn,west_germany).
country(frankfurt,east_germany).
country(rotterdan,netherlands).
country(tashkent,ussr).
country(pretoria,south_africa).
country(bucharest,romania).
country(budapest,hungary).
country(vienna,austria).
country(bern,switzerland).
country(geneva,switzerland).
country(zurich,switzerland).
country(bangkok,thailand).
country(seattle,usa).
country(tahiti,unknown).
country(saigon,vietnam).
country(yokohama,japan).
country(pnom_penh,cambodia).
country(panama_canal,panama).
country(naples,italy).
country(honolulu,usa).
country(berlin,east_germany).
country(helsinki,finland).
country(reykjavik,iceland).
country(thule,greenland).
country(godthab,greenland).
country(kabul,afghanistan).
country(wigan,uk).
country(damascus,syria).
country(jerusalem,israel).
country(beirut,lebanon).
country(amman,jordan).
country(kuala_lumpur,malaysia).
country(lima,peru).
country(quito,ecuador).
country(la_paz,bolivia).
country(asuncion,paraguay).
country(montevideo,uruguay).
country(suez_canal,egypt).
country(cairo,egypt).
country(tripoli,libya).
country(tunis,tunisia).
country(algiers,algeria).
country(rabat,morocco).
country(nicosia,cyprus).
country(riyadh,saudi_arabia).
country(baghdad,iraq).
country(manila,philippines).
country(taipei,taiwan).
country(vientiane,laos).
country(shanghai,china).
country(chunking,china).
country(canton,china).
country(mukden,china).
country(bombay,india).
country(madras,india).
country(calcutta,india).
country(dacca,bangladesh).
country(kuwait,kuwait).
country(doha,qatar).
country(samarkand,ussr).
country(addis_ababa,ethiopia).
country(mogadiscio,somalia).
country(kampala,uganda).
country(khartoum,sudan).
country(sana,yemen_arab_republic).
country(nairobi,kenya).
country(tananarive,madagascar).
country(durban,south_africa).
country(cape_town,south_africa).
country(windhoek,namibia).
country(luanda,angola).
country(kinshasa,zaire).
country(brazzaville,congo).
country(usambura,burundi).
country(kigali,rwanda).
country(libreville,gabon).
country(yaounde,cameroon).
country(fort_lamy,chad).
country(bangui,central_african_empire).
country(lagos,nigeria).
country(porto_novo,benin).
country(lome,togo).
country(accra,ghana).
country(ouagadougou,upper_volta).
country(niamey,niger).
country(bamako,mali).
country(monrovia,liberia).
country(freetown,sierra_leone).
country(conakry,guinea).
country(bathurst,gambia).
country(dakar,senegal).
country(nouakchott,mauritania).
country(salisbury,zimbabwe_rhodesia).
country(lourenco_marques,mozambique).
country(darwin,australia).
country(lusaka,zambia).
country(timbuktu,mali).
country(mexico_city,mexico).
country(guatemala_city,guatemala).
country(tegucigalpa,honduras).
country(managua,nicaragua).
country(guantanamo,cuba).
country(kingston,jamaica).
country(port_au_prince,haiti).
country(santo_domingo,dominican_republic).
country(san_jose,costa_rica).
country(panama_city,panama).
country(san_salvador,el_salvador).
country(greenwich,uk).
country(omaha,usa).
country(denver,usa).
country(nassau,bahamas).
country(manama,bahrain).
country(bridgetown,barbados).
country(thimphu,bhutan).
```

```
country(gaborone,botswana).
country(bujumbura,burundi).
country(praia,cape_verde).
country(bogota,colombia).
country(moroni,comoro_islands).
country(djibouti,djibouti).
country(suva,fiji).
country(saint_georges,grenada).
country(bissau,guinea_bissau).
country(georgetown,guyana).
country(abidjan,ivory_coast).
country(maseru,lesotho).
country(vaduz,liechtenstein).
country(luxembourg,luxembourg).
country(blantyre,malawi).
country(male,maldives).
country(valetta,malta).
country(port_louis,mauritius).
country(monte_carlo,monaco).
country(ulan_bator,mongolia).
country(yaren,nauru).
country(katmandu,nepal).
country(muscat,oman).
country(port_moresby,papua_new_guinea).
country(san_marino,san_marino).
country(sao_tome,sao_tome_and_principe).
country(victoria,seychelles).
country(honiara,solomon_islands).
country(colombo,sri_lanka).
country(paramaribo,surinam).
country(mbabane,swaziland).
country(dar_es_salaam,tanzania).
country(nuku_alofa,tonga).
country(port_of_spain,trinidad_and_tobago).
country(abu_dhabi,united_arab_emirates).
country(apia,western_samoa).
country(aden,yemen).
country(andorra_la_vella,andorra).
country(cayenne,french_guiana).
country(pisa,italy).
country(gangtok,sikkim).

/*
[8-1:] Get the first clause with primary key.
       do "q81(1000)." for one thousand iterations.
*/

q81(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q81(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q81(N,N) :- !.
loop_q81(I,N) :-
     I1 is I+1, country(moscow,X), !, loop_q81(I1,N).

loop_dummy(N,N) :- !.
loop_dummy(I,N) :-
     I1 is I+1, !, loop_dummy(I1,N).

/*
[8-2:] Get the first clause.
       do "q82(1000)." for one thousand iterations.
*/

q82(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q82(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q82(N,N) :- !.
loop_q82(I,N) :-
     I1 is I+1, country(X,ussr), !, loop_q82(I1,N).

/*
[8-3:] Get the last clause with primary key.
       do "q83(1000)." for one thousand iterations.
*/

q83(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q83(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q83(N,N) :- !.
loop_q83(I,N) :-
     I1 is I+1, country(gangtok,X), !, loop_q83(I1,N).

/*
[8-4:] Get the last clause.
       do "q84(1000)." for one thousand iterations.
*/
```

```
q84(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q84(0,N),
     statistics(runtime,[_,T1]), ,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q84(N,N) :- !.
loop_q84(I,N) :-
     I1 is I+1, country(X,sikkim), !, loop_q84(I1,N).

/*
[8-5:] Get the middle clause with primary key.
       do "q85(1000)." for one thousand iterations.
*/

q85(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q85(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q85(N,N) :- !.
loop_q85(I,N) :-
     I1 is I+1, country(manila,X), !, loop_q85(I1,N).

/*
[8-6:] Get the middle clause.
       do "q86(1000)." for one thousand iterations.
*/

q86(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q86(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q86(N,N) :- !.
loop_q86(I,N) :-
     I1 is I+1, country(X,philippines), !, loop_q86(I1,N).
```

```
% <OKUNO>BNCH3.PL.2,  7-Jul-84 11:40:50, Edit by OKUNO
% [9] **** Naive Reverse ****

:- public q91/1, nreverse/2, concatenate/3, list30/1.

/*
To optimize the compiled code, add the next declarations:

:- mode nreverse(+,-), concatenate(+,+,-), list30(-).
:- mode q91(-).
:- fastcode.
:- compactcode.

and also replace the definition of nreverse and concatenate by:

nreverse([X L0],L) :- !, nreverse(L0,L1), concatenate(L1,[X],L).
nreverse([],[]).

concatenate([X|L1],L2,[X|L3]) :- !, concatenate(L1,L2,L3).
concatenate([],L,L).

*/

nreverse([X|L0],L) :- nreverse(L0,L1), concatenate(L1,[X],L).
nreverse([],[]).

concatenate([X|L1],L2,[X|L3]) :- concatenate(L1,L2,L3).
concatenate([],L,L).

list30( [ 1, 2, 3, 4, 5, 6, 7, 8, 9,10,
          11,12,13,14,15,16,17,18,19,20,
          21,22,23,24,25,26,27,28,29,30 ]).

/*
[9-1:] Reverse a list of 30 elements.
       This comptation involves 496 LI (Logical Inference).
        do "q91(10)." for ten iterations.
*/

q91(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q91(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_list30(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q91(N,N) :- !.
loop_q91(I,N) :-
     I1 is I+1, list30(L), nreverse(L,X), !, loop_q91(I1,N).

loop_list30(N,N) :- !.
loop_list30(I,N) :-
     I1 is I+1, list30(L), !, loop_list30(I1,N).
```



```
% <OKUNO>BNCH4.PL.2,  7-Jul-84 11:55:34, Edit by OKUNO

% [10] **** Quick sort ****

:- public q101/1, qsort/3, partition/4, list50/1.

/*
To optimize the compiled code, add the next declarations:

:- mode qsort(+,-,+), partition(+,+,-,-), list50(-).
:- mode q101(-).
:- fastcode.
:- compactcode.

and also replace the definition of qsort and partition by:

qsort([X|L],R,R0) :- !,
          partition(L,X,L1,L2),
          qsort(L2,R1,R0),
          qsort(L1,R,[X|R1]).
qsort([],R,R).

partition([X|L],Y,[X|L1],L2) :- X =< Y, !, partition(L,Y,L1,L2).
partition([X|L],Y,L1,[X|L2]) :- !, partition(L,Y,L1,L2).
partition([],_,[],[]).

*/

qsort([X L],R,R0) :-
          partition(L,X,L1,L2),
          qsort(L2,R1,R0),
          qsort(L1,R,[X|R1]).
qsort([],R,R).

partition([X|L],Y,[X|L1],L2) :- X =< Y, !, partition(L,Y,L1,L2).
partition([X|L],Y,L1,[X|L2]) :-  partition(L,Y,L1,L2).
partition([],_,[],[]).

list50( [27,74,17,33,94,18,46,83,65, 2,
         32,53,28,85,99,47,28,82, 6,11,
         55,29,39,81,90,37,10, 0,66,51,
          7,21,85,27,31,63,75, 4,95,99,
         11,28,61,74,18,92,40,53,59, 8 ]).

/*
[10-1:] Sort 50 elements.
        The complexity of this computation is 609 LI.
        do "q101(10)." for ten iterations.
*/

q101(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q101(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_list50(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.
```

```
loop_ql0l(N,N) :- !.
loop_ql0l(I,N) :-
     Il is I+1, list50(L), qsort(L,X,[]), !, loop_ql0l(Il,N).

loop_list50(N,N) :- !.
loop_list50(I,N) :-
     Il is I+1, list50(L), !, loop_list50(Il,N).
```



```
% <OKUNO>BNCH5.PL.2,  7-Jul-84 11:59:06, Edit by OKUNO

% [11] **** Tree traversing ****

/* Bigapp : A non-determinate computation involving structure
   accessing and lots of backtracking.  Written by Paul F. Wilk.
*/

:- public ql11/1, conslist/2, concat/3, bigapp/1.
:- public ql12/1.

/*
To optimize the compiled code, add the next declarations:

:- mode conslist(+,-), concat(-,-,+), bigapp(+).
:- mode ql11(-), ql12(-).
:- fastcode.
:- compactcode.
*/

conslist(0 ,[]).
conslist(N,[1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5|L]) :-
        N > 0,
        N1 is N - 25,
        conslist(N1,L).

concat([],L,L).
concat([X|L1],L2,[X|L3]) :- concat(L1,L2,L3).

bigapp(L) :- concat(L1,L2,L), fail.
bigapp(_).

/*
[11-1:] Measure the time of consing 1000 elements by 25 elements.
        do "ql11(100)." for one hundred iterations.
*/

ql11(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_ql11(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     Gt1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql11(N,N) :- !.
loop_ql11(I,N) :-
     Il is I+1, conslist(1000,L), !, loop_ql11(Il,N).

loop_dummy(N,N) :- !.
loop_dummy(I,N) :-
     Il is I+1, !, loop_dummy(Il,N).

/*
[11-2:] Measure the time of decomposing a list of length 1000.
        do "ql12(1)." for only once.
```

```
*/

ql12(N) :-
     conslist(1000,L), assert(list1000(L)), ',
     statistics(garbage_collection,[_,_|Gl]),!,
     statistics(runtime,[_,_]),!,
     loop_ql12(0,N),
     statistics(runtime,[_,Tl]),',
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_list1000(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql12(N,N) :- !.
loop_ql12(I,N) :-
     I1 is I+1, list1000(L), !, bigapp(L), loop_ql12(I1,N).

loop_list1000(N,N) :- !.
loop_list1000(I,N) :-
     I1 is I+1, list1000(L), !, loop_list1000(I1,N).
```



```
% <OKUNO>BNCH6.PL.2,  7-Jul-84 12:04:25, Edit by OKUNO

% [12] **** Slow reverse ****

:- public srev/2, sapp/3.
:- public ql21/1, ql22/1, ql23/1.

/*
To optimize the compiled code, add the next declarations:

:- mode srev(+,-), sapp(+,+,-).
:- mode ql21(-), ql22(-), ql23(-).
:- fastcode.
:- compactcode.

and also replace the definition of srev and sapp as follows:

srev([],[]) :- !.
srev([A|X],Z) :- srev(X,Y),sapp(Y,[A],Z).

sapp([],X,X) :- !.
sapp(X,Y,Z) :- srev(X,[A|RX]),
               srev(X,Dummy),  % necessary fo fairness with Lisp
               srev(RX,RRX),
               sapp(RRX,[A|Y],Z).

*/

srev([],[]).
srev([A|X],Z) :- srev(X,Y),sapp(Y,[A],Z).

sapp([],X,X).
sapp(X,Y,Z) :- srev(X,[A|RX]),
               srev(X,Dummy),  % necessary fo fairness with Lisp
               srev(RX,RRX),
               sapp(RRX,[A|Y],Z).

/*
[12-1:] Slow reverse of list of four elements.
        do "ql21(100)" for one hundred iterations.
*/

ql21(N) :-
     statistics(garbage_collection,[_,_|Gl]),!,
     statistics(runtime,[_,_]),!,
     loop_ql21(0,N),                          % main loop
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),                          % dummy loop
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql21(N,N) :- !.
loop_ql21(I,N) :-
     I1 is I+1, srev([1,2,3,4],X), !, loop_ql21(I1,N).

loop_dummy(N,N) :- !.
loop_dummy(I,N) :-
```

```
    I1 is I+1, !, loop_dummy(I1,N).

/*
[12-2:] Slow reverse of list of five elements.
        do "ql22(100)" for one hundred iterations.
*/

ql22(N) :-
    statistics(garbage_collection,[_,_|Gl]), ,
    statistics(runtime,[_,_]),!,
    loop_ql22(0,N),                              % main loop
    statistics(runtime,[_,T1]),!,
    statistics(garbage_collection,[_,_|G2]),!,
    statistics(runtime,[_,_]),!,
    loop_dummy(0,N),                             % dummy loop
    statistics(runtime,[_,T2]),
    statistics(garbage_collection,[_,_|G3]),!,
    G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
    G4 is Gt2 - Gt2 - Gt1 - Gt3,
    T3 is T1-T2-G4, Total is T1-T2,
    write('Total =  '), write(Total),
    write('ms,  runtime = '), write(T3),
    write('ms,  gctime =  '), write(G4),
    write('ms,   for '), write(N), write('  iterations.'), nl.

loop_ql22(N,N) :- !.
loop_ql22(I,N) :-
    I1 is I+1, srev([1,2,3,4,5],X), !, loop_ql22(I1,N).

/*
[12-3:] Slow reverse of list of six elements.
        do "ql23(1)" for only once.
*/

ql23(N) :-
    statistics(garbage_collection,[_,_|G1]),!,
    statistics(runtime,[_,_]),!,
    loop_ql23(0,N),                              % main loop
    statistics(runtime,[_,T1]),!,
    statistics(garbage_collection,[_,_|G2]),!,
    statistics(runtime,[_,_]),!,
    loop_dummy(0,N),                             % dummy loop
    statistics(runtime,[_,T2]),
    statistics(garbage_collection,[_,_ G3]),!,
    G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
    G4 is Gt2 + Gt2 - Gt1 - Gt3,
    T3 is T1-T2-G4, Total is T1-T2,
    write('Total = '), write(Total),
    write('ms,  runtime = '), write(T3),
    write('ms,  gctime = '), write(G4),
    write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql23(N,N) :- !.
loop_ql23(I,N) :-
    I1 is I+1, srev([1,2,3,4,5,6],X), !, loop_ql23(I1,N).
```



```
% [13] ******** Lisp interpreter in Prolog ********
%                    Micro Lisp in Prolog

:- public lisp/2.
:- public tarai3/1, fib20/1, reverse30/1.
:- public ql31/1, ql32/1, ql33/1.

/*
To optimize the compiled code, add the next declarations:

:- mode apply(+,+,+,-), eval(+,+,-), evcon(+,+,-), evlis(+,+,-).
:- mode bind(+,+,+,-), value(+,+,-), lisp(+,-).
:- mode tarai3(-), fib10(-), reverse30(-).
:- mode ql31(-), ql32(-), ql33(-).
:- fastcode.
:- compactcode.
*/

apply(S,car,[[X|_]|_],X) :- !.
apply(S,cdr,[[_|Y]|_],Y) :- !.
apply(S,atom,[[_|_]|_],[]) :- !.
apply(S,atom,[X|_],t) :- atomic(X).

/* If atomic is not available, use the following definition:
                atomic([X|Y]) :- !, fail.
                atomic(X).
*/

apply(S,cons,[X,Y|_],[X|Y]) :- !.
apply(S,eq,[X,X|_],t) :- !.
apply(S,eq,[X,Y|_],[]) :- neq(X,X).
apply(S,null,[[]|_],t) :- !.
apply(S,sub1,[X|_],Y) :- !, Y is X-1.
apply(S,add1,[X|_],Y) :- !, Y is X+1.
apply(S,plus,[X,Y|_],V) :- !, V is X+Y.
apply(S,difference,[X,Y|_],V) :- !, V is X-Y.
apply(S,times,[X,Y|_],V) :- !, V is X*Y.
apply(S,greaterp,[X,Y|_],t) :- X>Y, !.
apply(S,greaterp,[_,_|_],[]).
apply(S,zerop,[0|_],t) :- !.
apply(S,zerop,[_|_],[]).
apply(S,[lambda,X,Y|_],R,V) :- !, bind(S,X,R,S1), eval(S1,Y,V).
apply(S,[label,X,Y|_],R,V) :- !, bind(S,[X],[Y],S1), apply(S1,Y,R,V).
apply(S,A,R,V) :- value(S,A,W), apply(S,W,R,V), !.

eval(S,[quote,X|_],X) :- !.
eval(S,[cond|X],V) :- !, evcon(S,X,V).
eval(S,[X|Y],V) :- !, evlis(S,Y,W), apply(S,X,W,V).
eval(S,A,V) :- value(S,A,V), !.

evcon(S,[[X,Y|_]|_],V) :- eval(S,X,W), neq(W,[]), !, eval(S,Y,V).
evcon(S,[_|U],V) :- evcon(S,U,V), !.

evlis(S,[],[]) :- !.
evlis(S,[X|Y],[U|V]) :- !, eval(S,X,U), evlis(S,Y,V).

bind(S,[],[],S) :- !.
bind(S,[A|X],[V|Y],S1) :- bind([[A|V]|S],X,Y,S1), !.

value([[A|V]|S],A,V) :- !.
value([X|S],A,V) :- value(S,A,V), !.

neq(X,X) :- !, fail.
neq(X,Y).

lisp(X,V) :-
```

```
   eval([[]],[t|t],[f],
         [reverse|[lambda,[l],[rev,l,[]]]],
         [rev|[lambda,[l,m],
                  [cond,[[atom,l],m],
                        [t,[rev,[cdr,l],[cons,[car,l],m]]]]]]
        ],X,V).

% ---- Lisp programs ----

tarai3(V) :-
 lisp([[label,tarai,
          [lambda,[x,y,z],
              [cond,[[greaterp,x,y],
                     [tarai,[tarai,[sub1,x],y,z],
                            [tarai,[sub1,y],z,x],
                            [tarai,[sub1,z],x,y]]],
                    [t,y]]]],
       [quote,6],[quote,3],[quote,0]], V).

fib10(V) :-
 lisp([[label,fib,
          [lambda,[n],
             [cond,[[eq,n,[quote,1]],[quote,1]],
                   [[eq,n,[quote,2]],[quote,1]],
                   [t,[plus,[fib,[sub1,n]],[fib,[difference,n,[quote,2]]]]]
             ]]],
       [quote,10]], V).

reverse30(V) :-
 lisp([reverse,[quote,[1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,
                       1,2,3,4,5,6,7,8,9,0]]], V).

/*
[13-1:] TARAI-3
        do "ql31(1)." for one once.
*/

ql31(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_ql31(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql31(N,N) :- !.
loop_ql31(I,N) :-
     I1 is I+1, tarai3(V), !, loop_ql31(I1,N).

loop_dummy(N,N) :- !.
loop_dummy(I,N) :-
     I1 is I+1, !, loop_dummy(I1,N).

/*
[13-2:] Fibonacci number for 10
        do "ql32(1)." for one once.
*/

ql32(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_ql32(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql32(N,N) :- !.
loop_ql32(I,N) :-
     I1 is I+1, fib10(V), !, loop_ql32(I1,N).

/*
[13-3:] Reverse a list of 30 elements
        do "ql33(1)." for one once.
*/

ql33(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_ql33(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql33(N,N) :- !.
loop_ql33(I,N) :-
     I1 is I+1, reverse30(V), !, loop_ql33(I1,N).
```

```
% [14] **** Eight Queens ****

:- public queen/2, queen_8/0, queen_all/0.
:- public ql41/1, ql42/1.

/*
To optimize the compiled code, add the next declarations:

:- mode queen(+,-), generate(+,-), try(+,+,+,+,+,+), select(+,-,-).
:- mode notmem(+,+).
:- mode ql41(-), ql42(-).
:- fastcode.
:- compactcode.
*/

queen(N,L) :- generate(N,L1), try(N,L1,[],L,[],[]).

generate(0,[]) :- !.
generate(N,[N|L]) :- N1 is N-1, generate(N1,L).

try(_,[],L,L,_,_) :- !.
try(N,S,L1,L,C,D) :-
    select(S,A,S1), C1 is N+A, notmem(C1,C), D1 is N-A,
    notmem(D1,D), M1 is N-1, try(M1,S1,[A|L1],L,[C|C],[D1 D]).

select([A|L],A,L).
select([A|L],X,[A|L1]) :- select(L,X,L1).

notmem(_,[]) :- !.
notmem(A,[B|L]) :- A=\=B, notmem(A,L).

queen_8 :- queen(8,L).

queen_all :- queen(8,L), fail.
queen_all.

/*
[14-1.] Get the first solution.
        do "ql41(1)." for only once.
*/

ql41(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_ql41(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-I2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql41(N,N) :- !.
loop_ql41(I,N) :-
     I1 is I+1, queen_8, !, loop_ql41(I1,N).

loop_dummy(N,N) :- !.
loop_dummy(I,N) :-
     I1 is I+1, !, loop_dummy(I1,N).

/*
[14-2.] Get all the solution.
        do "ql42(1)." for only once.
*/

ql42(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_ql42(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql42(N,N) :- !.
loop_ql42(I,N) :-
     I1 is I+1, queen_all, !, loop_ql42(I1,N).
```

```
% [15] Differentiation
%           This program is the same as that written in Lisp.

:- public d/3, diff1/1, diff2/1.
:- public ql51/1, ql52/1.

/*
To optimize the compiled code, add the next declarations:

:- mode d(+,+,-), simp_plus(+,+,-), simp_diff(+,+,-).
:- mode simp_times(+,+,-), simp_expt(+,+,-).
:- mode diff1(-), diff2(-).
:- mode ql51(-), ql52(-).
:- fastcode.
:- compactcode.
*/

:-op(300,xfy,^).

d(U+V,X,DY) :- !, d(U,X,DU), d(V,X,DV), simp_plus(DU,DV,DY).
d(U-V,X,DY) :- !, d(U,X,DU), d(V,X,DV), simp_diff(DU,DV,DY).
d(U*V,X,DY) :- !, d(U,X,DU), d(V,X,DV),
   simp_time(DU,V,D1), simp_time(U,DV,D2), simp_plus(D1,D2,DY).
d(U^0,X,0) :- !.
d(U^1,X,DY) :- !, d(U,X,DY).
d(U^N,X,DY) :- !, N1 is N-1, simp_expt(U,N1,Y), d(U*Y,X,DY).
d(X,X,1) :-!.
d(C,X,0) :- atomic(C), C \== 0.

simp_plus(X,Y,Z) :- integer(X), integer(Y), !, Z is X+Y.
simp_plus(0,Y,Y) :- !.
simp_plus(X,0,X) :- !.
simp_plus(X,Y,X+Y).

simp_diff(X,Y,Z) :- integer(X), integer(Y), !, Z is X-Y.
simp_diff(0,Y,-1*Y) :- !.
simp_diff(X,0,X) :- !.
simp_diff(X,Y,X+V) :- integer(Y), !, V is -Y.
simp_diff(X,Y,X-Y).

simp_time(X,Y,Z) :- integer(X), integer(Y), !, Z is X*Y.
simp_time(_,0,0) :- !.
simp_time(0,_,0) :- !.
simp_time(X,1,X) :- !.
simp_time(1,Y,Y) :- !.
simp_time(X,Y,X*Y).

simp_expt(_,0,1) :- !.
simp_expt(X,1,X) :- !.
simp_expt(X,N,X^N).

diff1(DF) :- d(x^3+3*x^2+3*x+1,x,DG),d(DG,x,DF).

diff2(DF) :-
   d((x-1)^5,x,D1),d(D1,x,D2),d(D2,x,D3),d(D3,x,D4),d(D4,x,DF).

/*
[15-1:] D(x^3+3*x^2+3*x+1)/DX
        do "ql51(100)." for one hundred iterations.
*/

ql51(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_ql51(0,N),
     statistics(runtime,[_,T]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql51(N,N) :- !.
loop_ql51(I,N) :-
     I1 is I+1, diff1(DF), !, loop_ql51(I1,N).

loop_dummy(N,N) :- !.
loop_dummy(I,N) :-
     I1 is I+1, !, loop_dummy(I1,N).

/*
[15-2:] D^5((x-1)^5)/DX
        do "ql52(1)." for only once.
*/

ql52(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_ql52(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_ql52(N,N) :- !.
loop_ql52(I,N) :-
     I1 is I+1, diff2(DF), !, loop_ql52(I1,N).
```

```
% [16] **** Database Manipulations ****
%               This database is created by J. A. Robinson et al.

/*
If numerical values cannot be represented in your system,
chang the format.  For example, DEC-10 Prolog cannot
express more than 2^18 or floating numbers.
*/

% population(afghanistan,20900000). ===> 161 clauses.
% adjoins(canada,usa). ===> 313 clauses.
% open_water(atlantic_ocean). ===> 40 clauses.
% country(moscow,ussr). ===>220 clauses.
% region(canada,north_america). ===> 162 clauses.
% produces(ussr,oil,491,1975). ===> 252 clauses.
% belongs(canada,nato). ===> 177 clauses.

:- public population/2, adjoins/2, open_water/2, country/2.
:- public region/2, produces/2, belongs/2.
:- public iscountry/1, landlocked/1, bordres/1.
:- public border_sea/2, oil_production/2.
:- public db2/1, db3/1, db4/1, db5/1.
:- public db6/1, db7/1, db8/1, db9/1, db10/1.
:- public q161/1, q162/1, q163/1, q164/1, q165/1.
:- public q166/1, q167/1, q168/1, q169/1, q160/1.

/*
To optimize the compiled code, add the next declarations:

:- mode size(+,+).
:- mode db2(-), db3(-), db4(-), db5(-), db6(-).
:- mode db7(-), db8(-), db9(-), db10(-).
:- mode q161(-), q162(-), q163(-), q164(-), q165(-).
:- mode q166(-), q167(-), q168(-), q169(-), q160(-).
:- fastcode.
:- compactcode.
*/

db2(S) :- setof(C, country(C,japan), S).

db3(C) :- region(C,far_east), iscountry(C), landlocked(C).

db4(S) :- setof(C, db3(C), S).

db5(C) :- iscountry(C), setof(X, border_sea(C,X), S), size(S,2).

border_sea(C,X) :- borders(C,X), open_water(X).
size([_,_],2).

db6(T) :- setof(C, db5(C),T).

db7(C) :-
       population(india,Y), borders(C,mediterranean_sea), iscountry(C),
       borders(C,C1), iscountry(C1), borders(C1,C2),
       population(C2,X), X>Y.

db8(S) :- setof(C, db7(C), S).

db9(S) :- setof([C,P], oil_production(C,P), S).
oil_production(C,P) :-
  belongs(C,opec), \+belongs(C,arab_league), produces(C,oil,P,1975).

db10(X) :-
  open_water(X), borders(X,C), region(C,africa), iscountry(C),
  borders(X,C1), region(C1,europe), iscountry(C1).
```



```
/*
[16-1:] Which country's capital is Tokyo?
        do "q161(1000)." for one thousand iterations.
*/

q161(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q161(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q161(N,N) :- !.
loop_q161(I,N) :-
     I1 is I+1, country(tokyo,X), !, loop_q161(I1,N).

loop_dummy(N,N) :- !.
loop_dummy(I,N) :-
     I1 is I+1, !, loop_dummy(I1,N).

/*
[16-2:] What are the cities in Japan?
        do "q162(100)." for one hundred iterations.
*/

q162(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q162(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q162(N,N) :- !.
loop_q162(I,N) :-
     I1 is I+1, db2(C), !, loop_q162(I1,N).

/*
[16-3:] Which far-east countres is landlocked?
        do "q163(1)." for only once.
*/

q163(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
```

```
*/

q168(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q168(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q168(N,N) :- !.
loop_q168(I,N) :-
     I1 is I+1, db8(C), !, loop_q168(I1,N).

/*
[16-9:] What are the oil production figures for the non-Arab OPEC
countries in the year 1975?
        do "q169(10)." for ten iterations.
*/

q169(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q169(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q169(N,N) :- !.
loop_q169(I,N) :-
     I1 is I+1, db9(C), !, loop_q169(I1,N).

/*
[16-10:] Which is the ocean that borders African countries and that
borders European countries?
        do "q160(10)." for ten iterations.
*/

q160(N) :-
     statistics(garbage_collection,[_,_|G1]),!,
     statistics(runtime,[_,_]),!,
     loop_q160(0,N),
     statistics(runtime,[_,T1]),!,
     statistics(garbage_collection,[_,_|G2]),!,
     statistics(runtime,[_,_]),!,
     loop_dummy(0,N),
     statistics(runtime,[_,T2]),
     statistics(garbage_collection,[_,_|G3]),!,
     G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
     G4 is Gt2 + Gt2 - Gt1 - Gt3,
     T3 is T1-T2-G4, Total is T1-T2,
     write('Total = '), write(Total),
     write('ms,  runtime = '), write(T3),
     write('ms,  gctime = '), write(G4),
     write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q160(N,N) :- !.
loop_q160(I,N) :-
     I1 is I+1, db10(C), !, loop_q160(I1,N).
```

```
    loop_q163(0,N),
    statistics(runtime,[_,T1]),!,
    statistics(garbage_collection,[_,_|G2]),!,
    statistics(runtime,[_,_]),!,
    loop_dummy(0,N),
    statistics(runtime,[_,T2]),
    statistics(garbage_collection,[_,_|G3]),!,
    G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
    G4 is Gt2 + Gt2 - Gt1 - Gt3,
    T3 is T1-T2-G4, Total is T1-T2,
    write('Total = '), write(Total),
    write('ms,  runtime = '), write(T3),
    write('ms,  gctime = '), write(G4),
    write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q163(N,N) :- !.
loop_q163(I,N) :-
    I1 is I+1, db3(C), !, loop_q163(I1,N).

/*
[16-4:] List up all the far-east countres which are landlocked.
        do "q164(1)." for only once.
*/

q164(N) :-
    statistics(garbage_collection,[_,_|G1]),!,
    statistics(runtime,[_,_]),!,
    loop_q164(0,N),
    statistics(runtime,[_,T1]),!,
    statistics(garbage_collection,[_,_|G2]),!,
    statistics(runtime,[_,_]),!,
    loop_dummy(0,N),
    statistics(runtime,[_,T2]),
    statistics(garbage_collection,[_,_|G3]),!,
    G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
    G4 is Gt2 + Gt2 - Gt1 - Gt3,
    T3 is T1-T2-G4, Total is T1-T2,
    write('Total = '), write(Total),
    write('ms,  runtime = '), write(T3),
    write('ms,  gctime = '), write(G4),
    write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q164(N,N) :- !.
loop_q164(I,N) :-
    I1 is I+1, db4(C), !, loop_q164(I1,N).

/*
[16-5:] Which countries border two seas?
        do "q165(1)." for only once.
*/

q165(N) :-
    statistics(garbage_collection,[_,_|G1]),!,
    statistics(runtime,[_,_]),!,
    loop_q165(0,N),
    statistics(runtime,[_,T1]),!,
    statistics(garbage_collection,[_,_|G2]),!,
    statistics(runtime,[_,_]),!,
    loop_dummy(0,N),
    statistics(runtime,[_,T2]),
    statistics(garbage_collection,[_,_|G3]),!,
    G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
    G4 is Gt2 + Gt2 - Gt1 - Gt3,
    T3 is T1-T2-G4, Total is T1-T2,
    write('Total = '), write(Total),
    write('ms,  runtime = '), write(T3),
    write('ms,  gctime = '), write(G4),
    write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q165(N,N) :- !.
loop_q165(I,N) :-
    I1 is I+1, db5(C), !, loop_q165(I1,N).

/*
[16-6:] List up all the countries which border two seas.
        do "q166(1)." for only once.
*/

q166(N) :-
    statistics(garbage_collection,[_,_|G1]),!,
    statistics(runtime,[_,_]),!,
    loop_q166(0,N),
    statistics(runtime,[_,T1]),!,
    statistics(garbage_collection,[_,_|G2]),!,
    statistics(runtime,[_,_]),!,
    loop_dummy(0,N),
    statistics(runtime,[_,T2]),
    statistics(garbage_collection,[_,_|G3]),!,
    G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
    G4 is Gt2 + Gt2 - Gt1 - Gt3,
    T3 is T1-T2-G4, Total is T1-T2,
    write('Total = '), write(Total),
    write('ms,  runtime = '), write(T3),
    write('ms,  gctime = '), write(G4),
    write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q166(N,N) :- !.
loop_q166(I,N) :-
    I1 is I+1, db6(C), !, loop_q166(I1,N).

/*
[16-7:] Which country bordering the Mediterranean borders a country
that is bordered by a country whose population exceeds the population
of the United State?
        do "q167(1)." for only once.
*/

q167(N) :-
    statistics(garbage_collection,[_,_|G1]),!,
    statistics(runtime,[_,_]),!,
    loop_q167(0,N),
    statistics(runtime,[_,T1]),!,
    statistics(garbage_collection,[_,_|G2]),!,
    statistics(runtime,[_,_]),!,
    loop_dummy(0,N),
    statistics(runtime,[_,T2]),
    statistics(garbage_collection,[_,_|G3]),!,
    G1 = [Gt1], G2 = [Gt2], G3 = [Gt3],
    G4 is Gt2 + Gt2 - Gt1 - Gt3,
    T3 is T1-T2-G4, Total is T1-T2,
    write('Total = '), write(Total),
    write('ms,  runtime = '), write(T3),
    write('ms,  gctime = '), write(G4),
    write('ms,   for '), write(N), write(' iterations.'), nl.

loop_q167(N,N) :- !.
loop_q167(I,N) :-
    I1 is I+1, db7(C), !, loop_q167(I1,N).

/*
[16-8:] List up all the countries which hold 16-7.
        do "q168(1)." for only once.
```

```
% ******** Definition of Places database ********

open_water(atlantic_ocean).
open_water(pacific_ocean).
open_water(caribbea_sea).
open_water(gulf_of_mexico).
open_water(straits_of_magellan).
open_water(straits_of_gibraltar).
open_water(mediterranean_sea).
open_water(arctic_ocean).
open_water(india_ocean).
open_water(arabia_sea).
open_water(gulf_of_aden).
open_water(red_sea).
open_water(panama_canal).
open_water(persia_gulf).
open_water(gulf_of_oman).
open_water(bering_sea).
open_water(english_channel).
open_water(north_sea).
open_water(baltic_sea).
open_water(black_sea).
open_water(bay_of_bengal).
open_water(yellow_sea).
open_water(formosa_strait).
open_water(south_china_sea).
open_water(philippine_sea).
open_water(gulf_of_siam).
open_water(sea_of_japan).
open_water(korea_strait).
open_water(suez_canal).
open_water(straits_of_hormuz).
open_water(gulf_of_suez).
open_water(taiwa_straits).
open_water(bab_el_mandeb).
open_water(bosporus).
open_water(dardanelles).
open_water(aegea_sea).
open_water(sea_of_marmara).
open_water(skagerrak).
open_water(kattegat).
open_water(gulf_of_bothnia).

population(afghanistan,2090).
population(albania,269).
population(algeria,1850).
population(andorra,2).
population(angola,576).
population(argentina,2640).
population(australia,1422).
population(austria,750).
population(bahamas,23).
population(bahrain,28).
population(bangladesh,8250).
population(barbados,26).
population(belgium,984).
population(benin,338).
population(bhutan,123).
population(bolivia,611).
population(botswana,72).
population(brazil,11545).
population(bulgaria,884).
population(burma,3220).
population(burundi,410).
population(cambodia,809).
population(camercon,682).
population(canada,2344).
population(cape_verde,31).
population(central_africa_empire,200).
population(chad,429).
population(chile,1088).
population(china,88000).
population(taiwan,1667).
population(colombia,2580).
population(comoro_islands,37).
population(congo,150).
population(costa_rica,212).
population(cuba,959).
population(cyprus,70).
population(czechoslovakia,1515).
population(denmark,511).
population(djibouti,22).
population(dominica_republic,517).
population(ecuador,782).
population(egypt,3950).
population(el_salvador,434).
population(ethiopia,2970).
population(greenland,5).
population(fiji,62).
population(finland,475).
population(france,5325).
population(gabon,54).
population(gambia,56).
population(east_germany,1657).
population(west_germany,6125).
population(ghana,1065).
population(greece,930).
population(grenada,11).
population(guatemala,663).
population(guinea,477).
population(guinea_bissau,54).
population(guyana,82).
population(haiti,483).
population(honduras,290).
population(hungary,1070).
population(iceland,22).
population(india,64300).
population(indonesia,14700).
population(iran,3420).
population(iraq,1235).
population(ireland,324).
population(italy,5667).
population(ivory_coast,671).
population(jamaica,211).
population(japan,11495).
population(jordan,208).
population(kenya,1480).
population(north_korea,1700).
population(south_korea,3700).
population(kuwait,119).
population(laos,354).
population(lebanon,316).
population(lesotho,109).
population(liberia,185).
population(libya,260).
population(liechtenstein,2).
population(luxembourg,37).
population(madagascar,877).
population(malawi,553).
population(malaysia,1295).
population(maldives,14).
```

```
population(mali,615).
population(malta,33).
population(mauritania,140).
population(mauritius,89).
population(mexico,6695).
population(monaco,3).
population(mongolia,157).
population(morocco,1867).
population(mozambique,995).
population(nauru,1).
population(nepal,1342).
population(netherlands,1333).
population(new_zealand,313).
population(nicaragua,239).
population(niger,500).
population(nigeria,6665).
population(norway,405).
population(oman,85).
population(pakistan,7750).
population(panama,182).
population(papua_new_guinea,298).
population(paraguay,287).
population(peru,1700).
population(philippines,4640).
population(poland,3505).
population(portugal,1000).
population(qatar,20).
population(zimbabwe_rhodesia,695).
population(romania,2163).
population(rwanda,460).
population(sa_marino,2).
population(sao_tome_and_principe,8).
population(saudi_arabia,980).
population(senegal,539).
population(seychelles,6).
population(sierra_leone,347).
population(singapore,234).
population(solomo_islands,20).
population(somalia,344).
population(south_africa,2676).
population(spain,3673).
population(sri_lanka,1420).
population(sudan,1655).
population(surinam,46).
population(swaziland,52).
population(sweden,830).
population(switzerland,631).
population(syria,800).
population(tanzania,1).
population(thailand,4538).
population(togo,241).
population(tonga,10).
population(trinidad_and_tobago,112).
population(tunisia,640).
population(turkey,4312).
population(uganda,1277).
population(ussr,26075).
population(united_arab_emirates,65).
population(uk,5580).
population(england_and_wales,4912).
population(scotland,519).
population(northern_ireland,153).
population(usa,218520).
population(upper_volta,648).
population(uruguay,202).
population(venezuela,1315).
population(vietnam,4927).
population(yeme_arab_republic,730).
population(yemen,185).
population(yugoslavia,2190).
population(zaire,2715).
population(zambia,550).
population(israel,370).

iscountry(X):-country(Y,X).

landlocked(X):- iscountry(X), \+(landlocked_test(X)).
landlocked_test(X):-borders(X,Z),open_water(Z).

country(moscow,ussr).
country(novaya_zemlya,ussr).
country(sverdlovsk,ussr).
country(vladivostok,ussr).
country(gorki,ussr).
country(novosibirsk,ussr).
country(syracuse,usa).
country(new_york_city,usa).
country(ithaca,usa).
country(albany,usa).
country(sa_francisco,usa).
country(sa_diego,usa).
country(washington,usa).
country(boston,usa).
country(rome_ny,usa).
country(rome,italy).
country(paris,france).
country(london,uk).
country(dublin,ireland).
country(stockholm,sweden).
country(copenhagen,denmark).
country(amsterdam,netherlands).
country(brussels,belgium).
country(madrid,spain).
country(athens,greece).
country(ankara,turkey).
country(istanbul,turkey).
country(tirane,albania).
country(sofia,bulgaria).
country(belgrade,yugoslavia).
country(warsaw,poland).
country(prague,czechoslovakia).
country(lisbon,portugal).
country(tehran,iran).
country(delhi,india).
country(islamabad,pakistan).
country(tokyo,japan).
country(brisbane,australia).
country(canberra,australia).
country(wellington,new_zealand).
country(djakarta,indonesia).
country(singapore,singapore).
country(peking,china).
country(hanoi,vietnam).
country(seoul,south_korea).
country(pyong_yang,north_korea).
country(recife,brazil).
country(brasilia,brazil).
country(santiago,chile).
country(oslo,norway).
country(vancouver,canada).
country(ottawa,canada).
country(montreal,canada).
```

```
country(toronto,canada).
country(havana,cuba).
country(rio_de_janeiro,brazil).
country(sa_paolo,brazil).
country(buenos_aires,argentina).
country(tierra_del_fuego,argentina).
country(punta_arenas,chile).
country(caracas,venezuela).
country(sa_juan,usa).
country(tampa,usa).
country(rangoon,burma).
country(bonn,west_germany).
country(frankfurt,east_germany).
country(rotterdam,netherlands).
country(tashkent,ussr).
country(pretoria,south_africa).
country(bucharest,romania).
country(budapest,hungary).
country(vienna,austria).
country(bern,switzerland).
country(geneva,switzerland).
country(zurich,switzerland).
country(bangkok,thailand).
country(seattle,usa).
country(tahiti,unknown).
country(saigon,vietnam).
country(yokohama,japan).
country(pnom_penh,cambodia).
country(panama_canal,panama).
country(naples,italy).
country(honolulu,usa).
country(berlin,east_germany).
country(helsinki,finland).
country(reykjavik,iceland).
country(thule,greenland).
country(godthab,greenland).
country(kabul,afghanistan).
country(wigan,uk).
country(damascus,syria).
country(jerusalem,israel).
country(beirut,lebanon).
country(amman,jordan).
country(kuala_lumpur,malaysia).
country(lima,peru).
country(quito,ecuador).
country(la_paz,bolivia).
country(asuncion,paraguay).
country(montevideo,uruguay).
country(suez_canal,egypt).
country(cairo,egypt).
country(tripoli,libya).
country(tunis,tunisia).
country(algiers,algeria).
country(rabat,morocco).
country(nicosia,cyprus).
country(riyadh,saudi_arabia).
country(baghdad,iraq).
country(manila,philippines).
country(taipei,taiwan).
country(vientiane,laos).
country(shanghai,china).
country(chunking,china).
country(canton,china).
country(mukden,china).
country(bombay,india).
country(madras,india).
country(calcutta,india).
country(dacca,bangladesh).
country(kuwait,kuwait).
country(doha,qatar).
country(samarkand,ussr).
country(addisababa,ethiopia).
country(mogadiscio,somalia).
country(kampala,uganda).
country(khartoum,sudan).
country(sana,yeme_arab_republic).
country(nairobi,kenya).
country(tananarive,madagascar).
country(durban,south_africa).
country(cape_town,south_africa).
country(windhoek,namibia).
country(luanda,angola).
country(kinshasa,zaire).
country(brazzaville,congo).
country(usambura,burundi).
country(kigali,rwanda).
country(libreville,gabon).
country(yaounde,cameroon).
country(fort_lamy,chad).
country(bangui,central_africa_empire).
country(lagos,nigeria).
country(porto_novo,benin).
country(lome,togo).
country(accra,ghana).
country(ouagadougou,upper_volta).
country(niamey,niger).
country(bamako,mali).
country(monrovia,liberia).
country(freetown,sierra_leone).
country(conakry,guinea).
country(bathurst,gambia).
country(dakar,senegal).
country(nouakchott,mauritania).
country(salisbury,zimbabwe_rhodesia).
country(lourenco_marques,mozambique).
country(darwin,australia).
country(lusaka,zambia).
country(timbuktu,mali).
country(mexico_city,mexico).
country(guatemala_city,guatemala).
country(tegucigalpa,honduras).
country(managua,nicaragua).
country(guantanamo,cuba).
country(kingston,jamaica).
country(port_au_prince,haiti).
country(santo_domingo,dominica_republic).
country(sa_jose,costa_rica).
country(panama_city,panama).
country(sa_salvador,el_salvador).
country(greenwich,uk).
country(omaha,usa).
country(denver,usa).
country(nassau,bahamas).
country(manama,bahrain).
country(bridgetown,barbados).
country(thimphu,bhutan).
country(gaborone,botswana).
country(bujumbura,burundi).
country(praia,cape_verde).
country(bogota,colombia).
country(moroni,comoro_islands).
country(djibouti,djibouti).
```

```
country(suva,fiji).
country(saint_georges,grenada).
country(bissau,guinea_bissau).
country(georgetown,guyana).
country(abidjan,ivory_coast).
country(maseru,lesotho).
country(vaduz,liechtenstein).
country(luxembourg,luxembourg).
country(blantyre,malawi).
country(male,maldives).
country(valetta,malta).
country(port_louis,mauritius).
country(monte_carlo,monaco).
country(ula_bator,mongolia).
country(yaren,nauru).
country(katmandu,nepal).
country(muscat,oman).
country(port_moresby,papua_new_guinea).
country(sa_marino,sa_marino).
country(sao_tome,sao_tome_and_principe).
country(victoria,seychelles).
country(honiara,solomo_islands).
country(colombo,sri_lanka).
country(paramaribo,surinam).
country(mbabane,swaziland).
country(dar_es_salaam,tanzania).
country(nuku_alofa,tonga).
country(port_of_spain,trinidad_and_tobago).
country(abu_dhabi,united_arab_emirates).
country(apia,western_samoa).
country(aden,yemen).
country(andorra_la_vella,andorra).
country(cayenne,french_guiana).
country(gangtok,sikkim).
country(pisa,italy).

region(canada,north_america).
region(mexico,north_america).
region(usa,north_america).
region(argentina,south_america).
region(bolivia,south_america).
region(brazil,south_america).
region(chile,south_america).
region(colombia,south_america).
region(ecuador,south_america).
region(guyana,south_america).
region(paraguay,south_america).
region(peru,south_america).
region(surinam,south_america).
region(uruguay,south_america).
region(venezuela,south_america).
region(costa_rica,central_america).
region(el_salvador,central_america).
region(guatemala,central_america).
region(honduras,central_america).
region(nicaragua,central_america).
region(panama,central_america).
region(bahamas,caribbean).
region(barbados,caribbean).
region(cuba,caribbean).
region(dominica_republic,caribbean).
region(grenada,caribbean).
region(haiti,caribbean).
region(jamaica,caribbean).
region(trinidad_and_tobago,caribbean).
region(albania,europe).
region(andorra,europe).
region(austria,europe).
region(belgium,europe).
region(bulgaria,europe).
region(cyprus,europe).
region(czechoslovakia,europe).
region(denmark,europe).
region(finland,europe).
region(france,europe).
region(east_germany,europe).
region(west_germany,europe).
region(greece,europe).
region(hungary,europe).
region(iceland,europe).
region(ireland,europe).
region(italy,europe).
region(liechtenstein,europe).
region(luxembourg,europe).
region(malta,europe).
region(monaco,europe).
region(netherlands,europe).
region(norway,europe).
region(poland,europe).
region(portugal,europe).
region(romania,europe).
region(sa_marino,europe).
region(spain,europe).
region(sweden,europe).
region(switzerland,europe).
region(ussr,europe).
region(uk,europe).
region(yugoslavia,europe).
region(bahrain,middle_east).
region(iran,middle_east).
region(iraq,middle_east).
region(israel,middle_east).
region(jordan,middle_east).
region(kuwait,middle_east).
region(lebanon,middle_east).
region(oman,middle_east).
region(qatar,middle_east).
region(saudi_arabia,middle_east).
region(syria,middle_east).
region(turkey,middle_east).
region(united_arab_emirates,middle_east).
region(yemen,middle_east).
region(yeme_arab_republic,middle_east).
region(china,far_east).
region(taiwan,far_east).
region(japan,far_east).
region(north_korea,far_east).
region(south_korea,far_east).
region(mongolia,far_east).
region(philippines,far_east).
region(cambodia,south_east_asia).
region(indonesia,south_east_asia).
region(laos,south_east_asia).
region(malaysia,south_east_asia).
region(singapore,south_east_asia).
region(thailand,south_east_asia).
region(vietnam,south_east_asia).
region(afghanistan,south_asia).
region(bangladesh,south_asia).
region(bhutan,south_asia).
region(burma,south_asia).
region(india,south_asia).
```

```
region(maldives,south_asia).
region(nepal,south_asia).
region(pakistan,south_asia).
region(sri_lanka,south_asia).
region(australia,oceania).
region(fiji,oceania).
region(nauru,oceania).
region(new_zealand,oceania).
region(papua_new_guinea,oceania).
region(solomo_islands,oceania).
region(tonga,oceania).
region(western_samoa,oceania).
region(algeria,africa).
region(angola,africa).
region(benin,africa).
region(botswana,africa).
region(burundi,africa).
region(cameroon,africa).
region(cape_verde,africa).
region(gabon,africa).
region(libya,africa).
region(nigeria,africa).
region(central_africa_empire,africa).
region(chad,africa).
region(comoro_islands,africa).
region(congo,africa).
region(djibouti,africa).
region(egypt,africa).
region(ethiopia,africa).
region(gambia,africa).
region(ghana,africa).
region(guinea,africa).
region(guinea_bissau,africa).
region(ivory_coast,africa).
region(kenya,africa).
region(lesotho,africa).
region(liberia,africa).
region(madagascar,africa).
region(malawi,africa).
region(mali,africa).
region(mauritania,africa).
region(mauritius,africa).
region(morocco,africa).
region(mozambique,africa).
region(namibia,africa).
region(niger,africa).
region(zimbabwe_rhodesia,africa).
region(rwanda,africa).
region(sao_tome_and_principe,africa).
region(senegal,africa).
region(seychelles,africa).
region(sierra_leone,africa).
region(somalia,africa).
region(south_africa,africa).
region(sudan,africa).
region(swaziland,africa).
region(tanzania,africa).
region(togo,africa).
region(tunisia,africa).
region(uganda,africa).
region(upper_volta,africa).
region(zaire,africa).
region(zambia,africa).
region(spanish_sahara,africa).
region(french_guiana,south_america).
region(sikkim,south_asia).
```



```
produces(ussr,oil,491,1975).
produces(ussr,oil,353,1970).
produces(usa,oil,41399,1975).
produces(usa,oil,475,1970).
produces(saudi_arabia,oil,352,1975).
produces(saudi_arabia,oil,188,1970).
produces(united_arab_emirates,oil,80,1975).
produces(united_arab_emirates,oil,37,1970).
produces(nigeria,oil,88,1975).
produces(nigeria,oil,54,1970).
produces(mexico,oil,36,1975).
produces(mexico,oil,21,1970).
produces(libya,oil,71,1975).
produces(libya,oil,159,1970).
produces(kuwait,oil,105,1975).
produces(kuwait,oil,150,1970).
produces(iraq,oil,111,1975).
produces(iran,oil,191,1970).
produces(iraq,oil,76,1970).
produces(iran,oil,267,1975).
produces(indonesia,oil,64,1975).
produces(indonesia,oil,42,1970).
produces(venezuela,oil,122,1975).
produces(venezuela,oil,194,1970).
produces(usa,wheat,58999,1975).
produces(usa,rice,5,1975).
produces(ussr,wheat,66,1975).
produces(ussr,rice,2,1975).
produces(algeria,oil,45999,1975).
produces(algeria,oil,47,1970).
produces(argentina,oil,20,1975).
produces(argentina,oil,20,1970).
produces(australia,oil,20,1975).
produces(australia,oil,8,1970).
produces(austria,oil,2,1975).
produces(austria,oil,2,1970).
produces(bahrain,oil,39999,1975).
produces(bahrain,oil,3,1970).
produces(brazil,oil,8,1975).
produces(brazil,oil,8,1970).
produces(bulgaria,oil,0,1975).
produces(bulgaria,oil,1,1970).
produces(canada,oil,67,1975).
produces(canada,oil,60,1970).
produces(chile,oil,1,1975).
produces(chile,oil,1,1970).
produces(china,oil,24,1975).
produces(china,oil,20999,1970).
produces(taiwan,oil,0,1975).
produces(taiwan,oil,1,1970).
produces(colombia,oil,8,1975).
produces(colombia,oil,11,1970).
produces(cuba,oil,0,1975).
produces(cuba,oil,0,1970).
produces(ecuador,oil,8,1975).
produces(ecuador,oil,0,1970).
produces(egypt,oil,8,1975).
produces(egypt,oil,16,1970).
produces(france,oil,1,1975).
produces(france,oil,2,1970).
produces(east_germany,oil,1,1975).
produces(west_germany,oil,5,1975).
produces(east_germany,oil,1,1970).
produces(west_germany,oil,7,1970).
produces(hungary,oil,2,1975).
```

```
produces(hungary,oil,1,1970).
produces(india,oil,8,1975).
produces(india,oil,6,1970).
produces(israel,oil,4,1975).
produces(israel,oil,4,1970).
produces(italy,oil,1,1975).
produces(italy,oil,1,1970).
produces(japan,oil,0,1975).
produces(japan,oil,1,1970).
produces(malaysia,oil,4,1975).
produces(malaysia,oil,1,1970).
produces(netherlands,oil,1,1975).
produces(netherlands,oil,2,1970).
produces(new_zealand,oil,1,1975).
produces(new_zealand,oil,1,1970).
produces(norway,oil,9,1975).
produces(pakistan,oil,1,1975).
produces(pakistan,oil,1,1970).
produces(peru,oil,4,1975).
produces(peru,oil,4,1970).
produces(poland,oil,1,1975).
produces(poland,oil,0,1970).
produces(romania,oil,14,1975).
produces(romania,oil,13,1970).
produces(spain,oil,2,1975).
produces(spain,oil,0,1970).
produces(syria,oil,10,1975).
produces(syria,oil,4,1970).
produces(tunisia,oil,5,1975).
produces(tunisia,oil,4,1970).
produces(turkey,oil,39999,1975).
produces(turkey,oil,3,1970).
produces(uk,oil,1,1975).
produces(uk,oil,1,1970).
produces(yugoslavia,oil,4,1975).
produces(yugoslavia,oil,3,1970).
produces(zaire,oil,1,1975).
produces(afghanistan,wheat,3,1975).
produces(argentina,wheat,9,1975).
produces(australia,wheat,12,1975).
produces(austria,wheat,1,1975).
produces(bangladesh,wheat,0,1975).
produces(belgium,wheat,1,1975).
produces(brazil,wheat,2,1975).
produces(bulgaria,wheat,3,1975).
produces(canada,wheat,17780,1975).
produces(chile,wheat,1020C,1975).
produces(china,wheat,41029,1975).
produces(taiwan,wheat,0,1975).
produces(colombia,wheat,0,1975).
produces(czechoslovakia,wheat,4,1975).
produces(denmark,wheat,0,1975).
produces(ecuador,0,1975).
produces(egypt,wheat,23299,1975).
produces(ethiopia,wheat,1,1975).
produces(finland,wheat,1,1975).
produces(france,wheat,15410,1975).
produces(east_germany,wheat,2,1975).
produces(west_germany,wheat,7139$,1975).
produces(greece,wheat,2,1975).
produces(hungary,wheat,40700,1975).
produces(india,wheat,24,1975).
produces(iran,wheat,5,1975).
produces(iraq,wheat,1,1975).
produces(ireland,wheat,0,1975).
produces(israel,wheat,0,1975).
produces(italy,wheat,10,1975).
produces(japan,wheat,0,1975).
produces(north_korea,wheat,0,1975).
produces(south_korea,wheat,C,1975).
produces(mexico,wheat,3,1975).
produces(nepal,wheat,0,1975).
produces(netherlands,wheat,0,1975).
produces(new_zealand,wheat,0,1975).
produces(pakistan,wheat,7,1975).
produces(peru,wheat,0,1975).
produces(poland,wheat,5,1975).
produces(portugal,wheat,0,1975).
produces(romania,wheat,4,1975).
produces(south_africa,wheat,1,1975).
produces(spain,wheat,4,1975).
produces(sweden,wheat,1,1975).
produces(switzerland,wheat,0,1975).
produces(syria,wheat,1,1975).
produces(turkey,wheat,14,1975).
produces(uk,wheat,4,1975).
produces(uruguay,wheat,0,1975).
produces(yugoslavia,wheat,4,1975).
produces(afghanistan,rice,0,1975).
produces(argentina,rice,0,1975).
produces(australia,rice,9,1975).
produces(bangladesh,rice,19,1975).
produces(brazil,rice,7,1975).
produces(bulgaria,rice,0,1975).
produces(burma,rice,9,1975).
produces(burma,wheat,0,1975).
produces(cambodia,rice,9,1975).
produces(chile,rice,0,1975).
produces(china,rice,116,1975).
produces(taiwan,rice,2,1975).
produces(colombia,rice,1,1975).
produces(cuba,rice,0,1975).
produces(ecuador,rice,0,1975).
produces(egypt,rice,2,1975).
produces(france,rice,0,1975).
produces(greece,rice,0,1975).
produces(hungary,rice,0,1975).
produces(india,rice,74,1975).
produces(indonesia,rice,22,1975).
produces(iran,rice,1,1975).
produces(iraq,rice,0,1975).
produces(italy,rice,1,1975).
produces(japan,rice,17,1975).
produces(north_korea,rice,3,1975).
produces(south_korea,rice,6,1975).
produces(laos,rice,0,1975).
produces(madagascar,rice,1,1975).
produces(malaysia,rice,2,1975).
produces(mexico,rice,0,1975).
produces(nepal,rice,2,1975).
produces(pakistan,rice,3,1975).
produces(panama,rice,0,1975).
produces(peru,rice,0,1975).
produces(philippines,rice,6,1975).
produces(portugal,rice,0,1975).
produces(romania,rice,0,1975).
produces(spain,rice,0,1975).
produces(sri_lanka,rice,1,1975).
produces(thailand,rice,15,1975).
produces(turkey,rice,0,1975).
produces(uruguay,rice,0,1975).
produces(venezuale,rice,0,1975).
```

```
produces(vietnam,rice,12,1975).
produces(yugoslavia,rice,0,1975).
produces(usa,uranium,9,1976).
produces(canada,uranium,4,1976).
produces(south_africa,uranium,3,1976).
produces(france,uranium,2,1976).
produces(niger,uranium,1,1976).
produces(belgium,steel,11,1975).
produces(canada,steel,13,1975).
produces(argentia,steel,1,1975).
produces(australia,steel,8,1975).
produces(austria,steel,4,1975).
produces(brazil,steel,8,1975).
produces(bulgaria,steel,2,1975).
produces(chile,steel,0,1975).
produces(china,steel,29,1975).
produces(taiwan,steel,0,1975).
produces(colombia,steel,0,1975).
produces(cuba,steel,0,1975).
produces(czechoslovakia,steel,14,1975).
produces(denmark,steel,0,1975).
produces(egypt,steel,0,1975).
produces(finland,steel,1,1975).
produces(france,steel,21,1975).
produces(east_germany,steel,6,1975).
produces(west_germany,steel,40,1975).
produces(greece,steel,0,1975).
produces(hungary,steel,3,1975).
produces(india,steel,7,1975).
produces(ireland,steel,0,1975).
produces(israel,steel,0,1975).
produces(italy,steel,21,1975).
produces(japan,steel,102,1975).
produces(north_korea,steel,2,1975).
produces(south_korea,steel,2,1975).
produces(luxembourg,steel,4,1975).
produces(mexico,steel,5,1975).
produces(netherlands,steel,4,1975).
produces(norway,steel,0,1975).
produces(peru,steel,0,1975).
produces(poland,steel,14,1975).
produces(portugal,steel,0,1975).
produces(romania,steel,9,1975).
produces(south_africa,steel,6,1975).
produces(zimbabwe_rhodesia,steel,0,1975).
produces(spain,steel,11,1975).
produces(sweden,steel,5,1975).
produces(switzerland,steel,0,1975).
produces(tunisia,steel,0,1975).
produces(turkey,steel,1,1975).
produces(ussr,steel,141,1975).
produces(uk,steel,20,1975).
produces(usa,steel,105,1975).
produces(venezuela,steel,1,1975).
produces(yugoslavia,steel,2,1975).

belongs(canada,nato).
belongs(canada,commonwealth).
belongs(canada,oecd).
belongs(mexico,oas).
belongs(usa,oas).
belongs(usa,nato).
belongs(argentina,oas).
belongs(bolivia,oas).
belongs(brazil,oas).
belongs(chile,oas).
belongs(ecuador,oas).
belongs(ecuador,opec).
belongs(guyana,commonwealth).
belongs(paraguay,oas).
belongs(peru,oas).
belongs(surinam,oas).
belongs(uruguay,oas).
belongs(venezuela,oas).
belongs(venezuela,opec).
belongs(costa_rica,oas).
belongs(el_salvador,oas).
belongs(guatemala,oas).
belongs(honduras,oas).
belongs(nicaragua,oas).
belongs(panama,oas).
belongs(bahamas,commonwealth).
belongs(barbados,oas).
belongs(barbados,commonwealth).
belongs(dominica_republic,oas).
belongs(grenada,oas).
belongs(grenada,commonwealth).
belongs(haiti,oas).
belongs(jamaica,oas).
belongs(jamaica,commonwealth).
belongs(trinidad_and_tobago,oas).
belongs(trinidad_and_tobago,commonwealth).
belongs(austria,efta).
belongs(austria,oecd).
belongs(belgium,nato).
belongs(belgium,eec).
belongs(belgium,oecd).
belongs(bulgaria,warsaw_pact).
belongs(cyprus,commonwealth).
belongs(czechoslovakia,warsaw_pact).
belongs(denmark,nato).
belongs(denmark,eec).
belongs(denmark,oecd).
belongs(finland,efta).
belongs(finland,oecd).
belongs(france,nato).
belongs(france,eec).
belongs(france,oecd).
belongs(east_germany,warsaw_pact).
belongs(west_germany,nato).
belongs(west_germany,eec).
belongs(west_germany,oecd).
belongs(greece,nato).
belongs(greece,oecd).
belongs(hungary,warsaw_pact).
belongs(iceland,nato).
belongs(iceland,efta).
belongs(iceland,oecd).
belongs(ireland,eec).
belongs(ireland,oecd).
belongs(italy,nato).
belongs(italy,eec).
belongs(italy,oecd).
belongs(luxembourg,nato).
belongs(luxembourg,eec).
belongs(luxembourg,oecd).
belongs(malta,commonwealth).
belongs(netherlands,nato).
belongs(netherlands,eec).
belongs(netherlands,oecd).
belongs(norway,nato).
belongs(norway,efta).
```

```
belongs(norway,oecd).
belongs(poland,warsaw_pact).
belongs(portugal,nato).
belongs(portugal,efta).
belongs(portugal,oecd).
belongs(romania,warsaw_pact).
belongs(spain,oecd).
belongs(sweden,oecd).
belongs(sweden,efta).
belongs(switzerland,efta).
belongs(switzerland,oecd).
belongs(ussr,warsaw_pact).
belongs(uk,nato).
belongs(uk,cento).
belongs(uk,eec).
belongs(uk,commonwealth).
belongs(uk,oecd).
belongs(bahrain,arab_league).
belongs(iran,opec).
belongs(iran,cento).
belongs(iraq,opec).
belongs(iraq,arab_league).
belongs(jordan,arab_league).
belongs(kuwait,opec).
belongs(kuwait,arab_league).
belongs(lebanon,arab_league).
belongs(oman,arab_league).
belongs(qatar,opec).
belongs(qatar,arab_league).
belongs(saudi_arabia,opec).
belongs(saudi_arabia,arab_league).
belongs(syria,arab_league).
belongs(turkey,nato).
belongs(turkey,cento).
belongs(turkey,oecd).
belongs(united_arab_emirates,opec).
belongs(united_arab_emirates,arab_league).
belongs(yemen,arab_league).
belongs(yeme_arab_republic,arab_league).
belongs(japan,oecd).
belongs(philippines,asean).
belongs(indonesia,opec).
belongs(indonesia,asean).
belongs(malaysia,asean).
belongs(malaysia,commonwealth).
belongs(singapore,asean).
belongs(singapore,commonwealth).
belongs(thailand,asean).
belongs(bangladesh,commonwealth).
belongs(india,commonwealth).
belongs(pakistan,cento).
belongs(sri_lanka,commonwealth).
belongs(australia,commonwealth).
belongs(fiji,commonwealth).
belongs(new_zealand,commonwealth).
belongs(new_zealand,oecd).
belongs(papua_new_guinea,commonwealth).
belongs(solomo_islands,commonwealth).
belongs(tonga,commonwealth).
belongs(western_samoa,commonwealth).
belongs(algeria,oau).
belongs(algeria,opec).
belongs(algeria,arab_league).
belongs(angola,oau).
belongs(benin,oau).
belongs(benin,ecwas).
```



```
belongs(botswana,oau).
belongs(botswana,commonwealth).
belongs(burundi,oau).
belongs(cameroon,oau).
belongs(cameroon,ecwas).
belongs(cape_verde,oau).
belongs(gabon,opec).
belongs(gabon,oau).
belongs(libya,opec).
belongs(libya,oau).
belongs(libya,arab_league).
belongs(nigeria,oau).
belongs(nigeria,opec).
belongs(nigeria,commonwealth).
belongs(nigeria,ecwas).
belongs(tunisia,arab_league).
belongs(somalia,arab_league).
belongs(sudan,arab_league).
belongs(morocco,arab_league).
belongs(mauritania,arab_league).
belongs(djibouti,arab_league).
belongs(egypt,arab_league).
belongs(gambia,commonwealth).
belongs(ghana,commonwealth).
belongs(kenya,commonwealth).
belongs(lesotho,commonwealth).
belongs(malawi,commonwealth).
belongs(mauritius,commonwealth).
belongs(seychelles,commonwealth).
belongs(sierra_leone,commonwealth).
belongs(swaziland,commonwealth).
belongs(tanzania,commonwealth).
belongs(togo,commonwealth).
belongs(uganda,commonwealth).
belongs(zambia,commonwealth).

adjoins(canada,usa).
adjoins(mexico,usa).
adjoins(atlantic_ocean,canada).
adjoins(canada,pacific_ocean).
adjoins(atlantic_ocean,usa).
adjoins(caribbea_sea,usa).
adjoins(pacific_ocean,usa).
adjoins(guatemala,mexico).
adjoins(guatemala,pacific_ocean).
adjoins(el_salvador,guatemala).
adjoins(guatemala,honduras).
adjoins(caribbea_sea,guatemala).
adjoins(british_honduras,guatemala).
adjoins(british_honduras,mexico).
adjoins(british_honduras,caribbea_sea).
adjoins(mexico,pacific_ocean).
adjoins(arctic_ocean,canada).
adjoins(arctic_ocean,greenland).
adjoins(arctic_ocean,norway).
adjoins(arctic_ocean,ussr).
adjoins(atlantic_ocean,greenland).
adjoins(atlantic_ocean,iceland).
adjoins(atlantic_ocean,ireland).
adjoins(atlantic_ocean,uk).
adjoins(atlantic_ocean,france).
adjoins(atlantic_ocean,spain).
adjoins(atlantic_ocean,portugal).
adjoins(atlantic_ocean,straits_of_gibraltar).
adjoins(mediterranean_sea,straits_of_gibraltar).
adjoins(spain,straits_of_gibraltar).
```

```
adjoins(morocco,straits_of_gibraltar).
adjoins(atlantic,ocean,morocco).
adjoins(atlantic_ocean,morocco).
adjoins(atlantic_ocean,spanish_sahara).
adjoins(atlantic_ocean,mauritania).
adjoins(atlantic_ocean,senegal).
adjoins(atlantic_ocean,gambia).
adjoins(atlantic_ocean,sierra_leone).
adjoins(atlantic_ocean,liberia).
adjoins(atlantic_ocean,ivory_coast).
adjoins(atlantic_ocean,ghana).
adjoins(atlantic_ocean,togo).
adjoins(atlantic_ocean,benin).
adjoins(atlantic_ocean,nigeria).
adjoins(atlantic_ocean,cameroon).
adjoins(atlantic_ocean,gabon).
adjoins(atlantic_ocean,angola).
adjoins(atlantic_ocean,namibia).
adjoins(atlantic_ocean,south_africa).
adjoins(atlantic_ocean,straits_of_magellan).
adjoins(argentina,atlantic,ocean).
adjoins(argentina,straits_of_magellan).
adjoins(chile,straits_of_magellan).
adjoins(chile,pacific_ocean).
adjoins(argentina,chile).
adjoins(argentina,bolivia).
adjoins(argentina,paraguay).
adjoins(argentina,uruguay).
adjoins(argentina,brazil).
adjoins(atlantic_ocean,uruguay).
adjoins(atlantic_ocean,brazil).
adjoins(atlantic_ocean,french_guiana).
adjoins(atlantic_ocean,guyana).
adjoins(atlantic_ocean,surinam).
adjoins(atlantic_ocean,venezuela).
adjoins(atlantic_ocean,trinidad_and_tobago).
adjoins(atlantic_ocean,caribbea_sea).
adjoins(atlantic_ocean,gulf_of_mexico).
adjoins(caribbea_sea,gulf_of_mexico).
adjoins(atlantic_ocean,india_ocean).
adjoins(india_ocean,south_africa).
adjoins(india_ocean,pacific_ocean).
adjoins(india_ocmczambique).
adjoins(india_ocean,mozambique).
adjoins(india_ocean,madagascar).
adjoins(arabia_sea,india_ocean).
adjoins(arabia_sea,gulf_of_aden).
adjoins(gulf_of_aden,red_sea).
adjoins(suez_canal,red_sea).
adjoins(mediterranean_sea,suez_canal).
adjoins(gulf_of_oman,persia_gulf).
adjoins(gulf_of_oman,arabia_sea).
adjoins(bering_sea,ussr).
adjoins(bering_sea,usa).
adjoins(arctic_ocean,bering_sea).
adjoins(bering_sea,pacific_ocean).
adjoins(egypt,suez_canal).
adjoins(panama,panama_canal).
adjoins(caribbea_sea,panama_canal).
adjoins(atlantic_ocean,english_channel).
adjoins(english_channel,north_sea).
adjoins(netherlands,north_sea).
adjoins(netherlands,west_germany).
adjoins(belgium,netherlands).
adjoins(belgium,north_sea).
adjoins(belgium,luxembourg).
adjoins(belgium,west_germany).
adjoins(belgium,france).
adjoins(english,channel,france).
adjoins(france,luxembourg).
adjoins(france,west_germany).
adjoins(france,switzerland).
adjoins(france,mediterranean_sea).
adjoins(france,italy).
adjoins(france,spain).
adjoins(andorra,france).
adjoins(mediterranean_sea,spain).
adjoins(portugal,spain).
adjoins(andorra,spain).
adjoins(italy,switzerland).
adjoins(italy,yugoslavia).
adjoins(austria,italy).
adjoins(italy,mediterranean_sea).
adjoins(east_germany,west_germany).
adjoins(east_germany,poland).
adjoins(czechoslovakia,west_germany).
adjoins(czechoslovakia,east_germany).
adjoins(austria,west_germany).
adjoins(switzerland,west_germany).
adjoins(denmark,west_germany).
adjoins(north_sea,west_germany).
adjoins(baltic_sea,west_germany).
adjoins(baltic_sea,east_germany).
adjoins(baltic_sea,poland).
adjoins(baltic_sea,ussr).
adjoins(baltic_sea,finland).
adjoins(baltic_sea,sweden).
adjoins(baltic_sea,kattegat).
adjoins(kattegat,skagerrak).
adjoins(skagerrak,north_sea).
adjoins(kattegat,denmark).
adjoins(kattegat,sweden).
adjoins(skagerrak,denmark).
adjoins(skagerrak,norway).
adjoins(norway,atlantic_ocean).
adjoins(norway,sweden).
adjoins(north_sea,norway).
adjoins(finland,norway).
adjoins(norway,ussr).
adjoins(finland,sweden).
adjoins(finland,ussr).
adjoins(czechoslovakia,poland).
adjoins(poland,ussr).
adjoins(austria,switzerland).
adjoins(austria,czechoslovakia).
adjoins(austria,yugoslavia).
adjoins(austria,hungary).
adjoins(hungary,romania).
adjoins(hungary,yugoslavia).
adjoins(czechoslovakia,hungary).
adjoins(czechoslovakia,ussr).
adjoins(hungary,ussr).
adjoins(romania,ussr).
adjoins(bulgaria,ussr).
adjoins(bulgaria,romania).
adjoins(black_sea,bulgaria).
adjoins(bulgaria,greece).
adjoins(bulgaria,turkey).
adjoins(bulgaria,yugoslavia).
adjoins(mediterranean_sea,yugoslavia).
adjoins(albania,yugoslavia).
adjoins(albania,greece).
```

```
adjoins(mediterranean_sea,albania).
adjoins(black_sea,turkey).
adjoins(black_sea,romania).
adjoins(black_sea,ussr).
adjoins(greece,yugoslavia).
adjoins(turkey,ussr).
adjoins(mediterranean_sea,turkey).
adjoins(iran,turkey).
adjoins(iraq,turkey).
adjoins(syria,turkey).
adjoins(pacific_ocean,straits_of_magellan).
adjoins(red_sea,suez_canal).
adjoins(india,pakistan).
adjoins(india,nepal).
adjoins(china,india).
adjoins(burma,india).
adjoins(bangladesh,india).
adjoins(bhutan,india).
adjoins(arabia_sea,india).
adjoins(bay_of_bengal,india).
adjoins(india,india_ocean).
adjoins(bay_of_bengal,india_ocean).
adjoins(arabia_sea,pakistan).
adjoins(india,sikkim).
adjoins(bhutan,sikkim).
adjoins(nepal,sikkim).
adjoins(china,nepal).
adjoins(afghanistan,iran).
adjoins(afghanistan,ussr).
adjoins(afghanistan,pakistan).
adjoins(afghanistan,china).
adjoins(china,ussr).
adjoins(china,mongolia).
adjoins(mongolia,ussr).
adjoins(burma,china).
adjoins(china,laos).
adjoins(china,vietnam).
adjoins(china,bhutan).
adjoins(china,sikkim).
adjoins(china,pakistan).
adjoins(china,north_korea).
adjoins(china,yellow_sea).
adjoins(east_china_sea,yellow_sea).
adjoins(china,east_china_sea).
adjoins(east_china_sea,pacific_ocean).
adjoins(east_china_sea,formosa_strait).
adjoins(formosa_strait,south_china_sea).
adjoins(china,south_china_sea).
adjoins(philippine_sea,pacific_ocean).
adjoins(philippines,philippine_sea).
adjoins(philippines,south_china_sea).
adjoins(south_china_sea,vietnam).
adjoins(laos,vietnam).
adjoins(cambodia,gulf_of_siam).
adjoins(gulf_of_siam,south_china_sea).
adjoins(israel,dead_sea).
adjoins(jordan,dead_sea).
adjoins(ethiopia,sudan).
adjoins(ethiopia,kenya).
adjoins(ethiopia,somalia).
adjoins(ethiopia,gulf_of_aden).
adjoins(djibouti,ethiopia).
adjoins(djibouti,gulf_of_aden).
adjoins(djibouti,somalia).
adjoins(ethiopia,red_sea).
adjoins(egypt,sudan).
```



```
adjoins(suez_canal,gulf_of_suez).
adjoins(gulf_of_suez,red_sea).
adjoins(taiwan,taiwa_straits).
adjoins(taiwan,pacific_ocean).
adjoins(gulf_of_siam,thailand).
adjoins(laos,thailand).
adjoins(cambodia,laos).
adjoins(north_korea,south_korea).
adjoins(north_korea,ussr).
adjoins(north_korea,yellow_sea).
adjoins(sea_of_japan,north_korea).
adjoins(south_korea,yellow_sea).
adjoins(sea_of_japan,south_korea).
adjoins(korea_strait,sea_of_japan).
adjoins(korea_strait,east_china_sea).
adjoins(korea_strait,south_korea).
adjoins(japan,korea_strait).
adjoins(japan,pacific_ocean).
adjoins(east_china_sea,japan).
adjoins(denmark,north_sea).
adjoins(greece,mediterranean_sea).
adjoins(greece,mediterranean_sea).
adjoins(israel,mediterranean_sea).
adjoins(egypt,mediterranean_sea).
adjoins(libya,mediterranean_sea).
adjoins(tunisia,mediterranean_sea).
adjoins(algeria,mediterranean_sea).
adjoins(morocco,mediterranean_sea).
adjoins(malta,mediterranean_sea).
adjoins(libya,tunisia).
adjoins(libya,algeria).
adjoins(libya,niger).
adjoins(chad,libya).
adjoins(libya,sudan).
adjoins(libya,egypt).
adjoins(algeria,tunisia).
adjoins(algeria,morocco).
adjoins(algeria,mauritania).
adjoins(algeria,mali).
adjoins(algeria,niger).
adjoins(dahomey,nigeria).
adjoins(niger,nigeria).
adjoins(chad,nigeria).
adjoins(cameroon,nigeria).
adjoins(iran,ussr).
adjoins(caspia_sea,iran).
adjoins(iran,pakistan).
adjoins(iran,iraq).
adjoins(iran,persia_gulf).
adjoins(iran,gulf_of_oman).
adjoins(oman,gulf_of_oman).
adjoins(iraq,syria).
adjoins(iraq,jordan).
adjoins(iraq,saudi_arabia).
adjoins(iraq,kuwait).
adjoins(iraq,persia_gulf).
adjoins(iran,straits_of_hormuz).
adjoins(persia_gulf,straits_of_hormuz).
adjoins(jordan,israel).
adjoins(jordan,syria).
adjoins(jordan,saudi_arabia).
adjoins(jordan,gulf_of_aqabq).
adjoins(gulf_of_aqabq,red_sea).
adjoins(israel,egypt).
adjoins(israel,lebanon).
adjoins(israel,syria).
```

```
adjoins(israel,gulf_of_aqaba).
adjoins(dardanelles,turkey).
adjoins(dardanelles,sea_of_marmara).
adjoins(sea_of_marmara,turkey).
adjoins(sea_of_marmara,bosporus).
adjoins(bosporus,turkey).
adjoins(bab_el_mandeb,djibouti).
adjoins(bab_el_mandeb,yemen).
adjoins(bab_el_mandeb,yeme_arab_republic).
adjoins(bab_el_mandeb,red_sea).
adjoins(bab_el_mandeb,gulf_of_aden).
adjoins(straits_of_hormuz,united_arab_emirates).
adjoins(straits_of_hormuz,oman).
adjoins(straits_of_dover,uk).
adjoins(straits_of_dover,france).
adjoins(straits_of_dover,english_channel).
adjoins(straits_of_dover,north_soa).
adjoins(monaco,mediterranean_sea).
adjoins(arctic_ocean,usa).

borders(X,Y):-adjoins(X,Y).
borders(X,Y):-adjoins(Y,X).

% ******** That's all ********
```


Benchmark	Interpreted code (1)	Interpreted code (2)	Compiled code (3)	Compiled code (4)
[1-1:] Atom-1				
[1-2:] Atom-5				
[2-1:] Var-1				
[2-2:] Var-5				
[3-1:] Con-1				
[3-2:] Con-5				
[4-1:] Str-1				
[4-2:] Str-5				
[5-1:] Str-Var-1				
[5-2:] Str-Var-5				
[6-1:] Var-Str-1				
[6-2:] Var-Str-5				
[7-1:] Det-Call				
[7-2:] Ndet-Call				
[7-3:] Shallow-Back				
[7-4:] Deep-Back				
[8-1:] Key-First				
[8-2:] First				
[8-3:] Key-Last				
[8-4:] Last				
[8-5:] Key-Middle				
[8-6:] Middle				
[9-1:] Rev-30				
[10-1:] Sort-50				
[11-1:] Cons-1000				
[11-2:] Trav-1000				
[12-1:] Srev-4				
[12-2:] Srev-5				
[12-3:] Srev-6				
[12-1:] Lisp-Tarai-3				
[12-2:] Lisp-Fib-10				
[12-3:] Lisp-Rev-30				
[13-1:] 8-Queen-1				
[13-2:] 8-Queen-all				
[15-1:] Diff-1				
[15-2:] Diff-2				
[16-1:] DB-1				
[16-2:] DB-2				
[16-3:] DB-3				
[16-4:] DB-4				
[16-5:] DB-5				
[16-6:] DB-6				
[16-7:] DB-7				
[16-8:] DB-8				
[16-9:] DB-9				
[16-10:] DB-10				
[17-1:] Fib-Fact				

ECRC Prolog Benchmark Programs (summary)

J.C. SYRE
ECRC (European Computer-industry Research Center)
WEST GERMANY

```
*********************************************
*** BENCHMARK PROGRAMS FOR PROLOG SYSTEMS   ***
***     (FINAL VERSION)                    ***
*********************************************
```

Part 1 (of 3)

1. Simple Benchmark programs.

The phenomena we measure are:

- calls (section 1.1)
  program: boresea(N);
- non-determinism (section 1.2)
  programs: choice_point(N),
  choice_point0ar(N), baktrak1(N), baktrak2(N)
- handling of environments (section 1.3)
  programs: envir(N), envir0ar(N)
- indexing (section 1.4)
  program: index(N)
- unification (section 1.5)
  programs: construct_list(N), match_list(N),
  construct_structure(N), match_structure(N),
  match_nested_structure(N), general_unification(N)
- dereferencing (section 1.6)
  program: deref(N)
- cut (section 1.7)
  program: cuttest(N)

```
/* 1.1. Program to test calls  (boresea).                              */

/* This program is called with the query "?-boresea(X)."              */
/* X is the number of loop iterations executed. It should be big      */
/* enough to give significant results.                                */
/* suggested value for X: 100 for interpreted code*/
/*                        1000 for compiled code   */
/* average values for C-prolog interpreter:        */
/*       X=1000, Tloop=27.1 T.comp=1.0 Tnet=26.1 Klips=7.7 */

boresea(X)
     :- T1 is cputime,
        do_max_KLips(X),       /* calls the loop to execute the  */
        T2 is cputime,         /* sequence of 200 predicates     */
        compens_loop(X),       /* compensation loop              */
        T3 is cputime,
        print_times(T1,T2,T3,X,200)./*compute and print results  */

compens_loop(0).                       /* compensation loop       */
compens_loop(X) :- Y is X - 1, compens_loop(Y).

print_times(T1,T2,T3,X,I) :-           /* prints the results      */
        TT1 is T2 - T1,
        TT2 is T3 - T2,
        TT is TT1 - TT2,
        write('T overall loop:   '),write(TT1), nl,
        write('T compens loop:   '),write(TT2), nl,
        write('T net:            '),write(TT),nl,
        write('KLips:            '),
        Li is I * X,
        Lips is Li / TT,
        KLips is Lips / 1000,
        write(KLips),nl,nl.

do_max_KLips(0).           /* loop calling the actual benchmark */
do_max_KLips(X) :- lips1, Y is X - 1, do_max_KLips(Y).

/* predicates to test call */

lips1 :- lips2.
lips2 :- lips3.
   :
   :
lips199 :- lips200.
lips200.

/* 1.2. Program to test non deterministic behaviour                   */

/* The predicates are called:                                         */

/*    o  "choice_point(N)"     - creation of choice points          */
/*    o  "choice_point0ar(N)   - same, with 0 arg                   */
/*    o  "baktrak1(N)"         - deep backtracking                  */
/*    o  "baktrak2(N)"         - shallow backtracking               */

/*  N is the number of loop iterations executed  */

/* predicate to test creation of choice points without backtracking */
/* suggested value for N: 1000 */
/* results for  Cprolog N=1000 */
/* Tloop=5.95 Tcompens=0.98 Tnet=4.97 Klips=4.02 */

choice_point(N):-T1 is cputime,
        cre_CP(N), T2 is cputime,
        compens_loop(N), T3 is cputime,
        print_times(T1,T2,T3,N,20).

/* predicate choice_point, but with zero argument  */
/* suggested value for N: 1000 */
/* results for Cprolog: N=1000 */
/* Tloop=3.55 Tcompens=0.98 Tnet=2.57 Klips=7.7  */

choice_point0ar(N):-T1 is cputime,
        cre_CP0ar(N), T2 is cputime,
        compens_loop(N), T3 is cputime,
        print_times(T1,T2,T3,N,20).
/* Predicate to test the (deep) backtracking mechanism. */
/* suggested value for N: 1000 (interp), 2000(comp) */
/* results for Cprolog: N=1000  */
/* Tloop=9.63 Tcomp=1 Tnet=8.63 Klips=2.32  */

baktrak1(N)
     :- T1 is cputime,
        deep_back(N),
        T2 is cputime,
        compens_loop(N),
        T3 is cputime,
        print_times(T1,T2,T3,N,20).

/* Predicate to test the (shallow) backtracking mechanism */
/* suggested value for N: 1000 (interp), 2000 (comp) */
/* results for Cprolog: N=1000  */
/* Tloop=3.63  Tcomp=0.95 Tnet=2.68 Klips=7.45 */

baktrak2(X)
     :- T1 is cputime,
        shallow_back(X), T2 is cputime,
        compens_loop(X), T3 is cputime,
```

```
        print_times(T1,T2,T3,X,20).

/* compensation loop, used to measure the time spent in the loop  */
compens_loop(0).
compens_loop(X) :- Y is X - 1, compens_loop(Y).

/* loop to test choice point creation   */
cre_CP(0).
cre_CP(N):-M is N-1, ccp1(0,0,0), cre_CP(M).

cre_CP0ar(0).
cre_CP0ar(N):-M is N-1, ccp1, cre_CP0ar(M).

/* loop to test deep backtracking       */
deep_back(0).
deep_back(X) :- pd(_,_,_), Y is X - 1, deep_back(Y).

/* loop to test shallow backtracking */
shallow_back(0).
shallow_back(X) :- ps(_,a,b), Y is X - 1, shallow_back(Y).

print_times(T1,T2,T3,X,I) :-        /* prints the results */
        TT1 is T2 - T1,
        TT2 is T3 - T2,
        TT is TT1 - TT2,
        write(' T overall loop:   '),write(TT1), nl,
        write(' T compens loop:   '),write(TT2), nl,
        write('T net:             '),write(TT),nl,
        write('KLips:             '),
        Li is I * X,
        Lips is Li / TT,
        KLips is Lips / 1000,
        write(KLips),nl,nl.

/*  ccp1 creates 20 choice points */
/* ccp1 is the beginning of a set of predicates  */
/* ccpXX exists with 3 arguments, and 0 args. */

ccp1(X,Y,Z):-ccp2(X,Y,Z).
ccp1(X,Y,Z).
ccp2(X,Y,Z):-ccp3(X,Y,Z).
ccp2(X,Y,Z).
    :
    :
ccp19(X,Y,Z):-ccp20(X,Y,Z).
ccp19(X,Y,Z).

ccp20(X,Y,Z).
ccp20(X,Y,Z).

ccp1:-ccp2.
ccp1.
ccp2:-ccp3.
ccp2.
  :
  :
ccp19:-ccp20.
ccp19.

ccp20.
ccp20.

/*  deep backtracking */
/*  The call to pd creates a choice point, and invokes a      */
/*  call to q. It will fail and there will be a backtracking  */
/*  step  to try the next clause defining pd. pd has 21       */
/*  clauses,thus failure                                      */
/*  occurs 20 times                                           */

pd(X1,X2,_) :- q(X1,X2,a).      /* 1
pd(X1,X2,_) :- q(X1,X2,a).      /* 2
        :
        :
pd(X1,X2,_) :- q(X1,X2,a).      /* 19
pd(X1,X2,_) :- q(X1,X2,a).      /* 20
pd(X1,X2,_).

q(X1,X2,b).

/*   shallow backtracking */
/*   The ps predicate fails 20 times. The shallow backtracking   */
/*   will not restore all current state registers in Prolog      */
/*   systems which perform this optimisation, while others will. */

ps(_,X,X).      /* 1
ps(_,X,X).      /* 2
    :
    :
ps(_,X,X).      /* 19
ps(_,X,X).      /* 20
ps(_,_,_).
```

```
*********************************************
*** BENCHMARK PROGRAMS FOR PROLOG SYSTEMS  ***
***    (FINAL VERSION)                    ***
*********************************************
                Part 2 (of 3)

/* 1.3. Program to test the handling of environments  (envir).  */

/* envir(N): 3 arguments environment creation                    */
/*         creates 79 environments and 158 calls                 */
/* suggested value for N: 1000 (interp), 1000 (comp)             */
/* results for Cprolog: N=1000                                   */
/* Tloop=38.6 Tcomp=0.97 Tnet=37.6 Klips=4.23                    */

envir(N):-T1 is cputime,
        cre_env(N), T2 is cputime,
        compens_loop(N), T3 is cputime,
        print_times(T1,T2,T3,N,159).

cre_env(0).
cre_env(N):-M is N-1, env0(X,Y,Z), cre_env(M).

compens_loop(0).
compens_loop(N):-M is N-1,compens_loop(M).

env0(X,Y,Z):-env1(Z,X,Y),env2(Y,Z,X). /* creates 79 environments */
env1(X,Y,Z):-env3(Z,Y,X),env4(Y,Z,X).
env2(X,Y,Z):-env3(Z,Y,X),env4(Y,Z,X).  /* and 158 calls */
env3(X,Y,Z):-env5(Z,Y,X),env6(Y,Z,X).
env4(X,Y,Z):-env5(Z,Y,X),env6(Y,Z,X).
env5(X,Y,Z):-env7(Z,Y,X),env8(Y,Z,X).
env6(X,Y,Z):-env7(Z,Y,X),env8(Y,Z,X).
env7(X,Y,Z):-env9(Z,Y,X),env10(Y,Z,X).
env8(X,Y,Z):-env9(Z,Y,X),env10(Y,Z,X).
env9(X,Y,Z):-env11(Z,Y,X),env12(Y,Z,X).
env10(X,Y,Z):-env12(Z,Y,X),env12(Y,Z,X).
env11(X,Y,Z):-env12(Z,Y,X),env12(Y,Z,X).
env12(X,Y,Z).

/* envir0ar(N): zero argument environment creation */
/*       creates 79 environments and 158 calls     */
/* suggested value for N: 1000 (interp), 1300 (comp) */
/* results for Cprolog: N=1000                       */
/* Tloop=18.88 Tcomp=1.01 Tnet=17.87 Klips=8.9 */

envir0ar(N):-T1 is cputime,
        cre_env0ar(N), T2 is cputime,
        compens_loop(N), T3 is cputime,
        print_times(T1,T2,T3,N,159).

cre_env0ar(0).
cre_env0ar(N):-M is N-1, env0, cre_env(M).

env0:-env1,env2. /* creates 79 environments */
env1:-env3,env4.
env2:-env3,env4.  /* and 158 calls */
env3:-env5,env6.
env4:-env5,env6.
env5:-env7,env8.
env6:-env7,env8.
env7:-env9,env10.
env8:-env9,env10.
env9:-env11,env12.
env10:-env12,env12.
env11:-env12,env12.
env12.
```



```
print_times(T1,T2,T3,X,I) :-        /* prints the results */
        TT1 is T2 - T1,
        TT2 is T3 - T2,
        TT is TT1 - TT2,
        write('T overall loop:   '),write(TT1), nl,
        write('T compens loop:   '),write(TT2), nl,
        write('T net:            '),write(TT).nl,
        write('KLips:            '),
        Li is I * X,
        Lips is Li / TT,
        KLips is Lips / 1000,
        write(KLips),nl,nl.

/* 1.4. Program to test indexing mechanisms.                          */

/* This program is called with "index(N)"                             */
/* It tests the efficiency of simple indexing on the 1st argument     */
/* suggested value for N: 500 (interp), 2000(comp)                    */
/* results for Cprolog: N=500                                         */
/* Tloop=8.98 Tcomp=0.52 Tnet=8.47 Klips=1.24                         */

index(N)
     :- T1 is cputime,
        index_loop(N), T2 is cputime,
        compens_loop(N), T3 is cputime,
        print_times(T1,T2,T3,N,21).

/* loop with calls to the actual benchmark program for indexing */
index_loop(0).
index_loop(X) :- p(a), p([a]), p(s(a)), /* queries to the actual */
                 p(b), p([b]), p(t(b)), /* benchmark program     */
                 p(c), p([c]), p(u(c)),
                 p(d), p([d]), p(v(d)),
                 p(e), p([e]), p(w(e)),
                 p(f), p([f]), p(x(f)),
                 p(g), p([g]), p(y(g)),
                 Y is X - 1, index_loop(Y).

/* compensation loop */
compens_loop(0).
compens_loop(X) :- Y is X - 1, compens_loop(Y).

/* test program which can be optimised by indexing */
p(a).
p([a]).
p(s(a)).
p(b).
p([b]).
p(t(b)).
p(c).
p([c]).
p(u(c)).
p(d).
p([d]).
p(v(d)).
p(e).
p([e]).
p(w(e)).
p(f).
p([f]).
p(x(f)).
p(g).
p([g]).
p(y(g)).
```

```
print_times(T1,T2,T3,X,I) :-        /* prints the results */
        TT1 is T2 - T1,
        TT2 is T3 - T2,
        TT is TT1 - TT2,
        write('T first loop:     '),write(TT1), nl,
        write('T compens loop:   '),write(TT2), nl,
        write('T net:            '),write(TT),nl,
        write('KLips:            '),
        Li is I * X,
        Lips is Li / TT,
        KLips is Lips / 1000,
        write(KLips),nl,nl.

/* 1.5. Programs to test unification.                              */

/* The predicates are called with:                                 */
/*            benchmark_name(X).                                   */
/* where benchmark_name is the name of the predicate              */
/*       and X is the number of (external) loop iterations        */
/* The benchmarks on this file are:   construct_list              */
/*                                    match_list                  */
/*                                    construct_structure         */
/*                                    match_structure             */
/*                                    match_nested_structure      */
/*                                    general_unification         */

/*  Test of list construction via unification                      */
/* suggested value for N: 100 (interp), 500(comp)                  */
/* results for Cprolog: N=100                                      */
/* Tloop=2.49 Tcomp=0.08 Tnet=2.41 Klips=4.2                       */
construct_list(X)              /* uses unification to    */
     :- T1 is cputime,         /* construct a list of 100 elts */
        do_construct_list(X),
        T2 is cputime,
        compens_loop(X),
        T3 is cputime,
        print_times(T1,T2,T3,X,100).

/*  Test of list matching unification                              */
/* suggested value for N: 100 (interp), 1000(comp)                 */
/* results for Cprolog: N=100                                      */
/* Tloop=4.55 Tcomp=0.1 Tnet=4.46 Klips=2.2                        */
match_list(X)
     :- list100(Z), /* construction of the matching list is done  */
        T1 is cputime,      /* outside of the loop,     */
        do_match_list(X,Z), /* in order not to coun     */
        T2 is cputime,      /* construction of the list */
        compens_loop(X),
        T3 is cputime,
        print_times(T1,T2,T3,X,100).

/*  Test of structure construction via unification                 */

/* this program is equivalent to construct_list, except           */
/* that it uses the standard structure representation             */
/*instead of the simplified list notation                         */
/* suggested value for N: 100 (interp), 500(comp)                 */
/* results for Cprolog: N=100                                      */
/* Tloop=2.56 Tcomp=0.08 Tnet=2.48 Klips=4                         */
construct_structure(X)
     :- T1 is cputime,
        do_construct_structure(X),
        T2 is cputime,
        compens_loop(X),
        T3 is cputime,
        print_times(T1,T2,T3,X,100).
```



```
/*  Test of structure matching via unification    */

/* this predicate matches a list of 100 elements  */
/* in structure notation                          */
/* suggested value for N: 100 (interp), 100(comp) */
/* results for Cprolog: N=100                     */
/* Tloop=4.66 Tcomp=0.1 Tnet=4.56 Klips=2.2       */
match_structure(X)
     :- structure100(Z),
        T1 is cputime,
        do_match_structure(X,Z),
        T2 is cputime,
        compens_loop(X),
        T3 is cputime,
        print_times(T1,T2,T3,X,100).

/*  Test to match a nested structure              */

/* this predicate tests the (compiled) unification */
/* of a complex structure                         */
/* suggested value for N: 200 (interp), 200(comp) */
/* results for Cprolog: N=200                     */
/* Tloop=1.34 Tcomp=0.17 Tnet=1.18 Klips=0.17     */
match_nested_structure(X)
     :- nested_structure1(Z), /* the structure to match is     */
                              /* constructed outside the loop, */
                              /* in order to measure           */
                              /* only the matching time        */
        T1 is cputime,
        do_match_nested_structure(X,Z),
        T2 is cputime,
        compens_loop(X),
        T3 is cputime,
        print_times(T1,T2,T3,X,1).

/*     Test of general unification of 2 complex structures     */

/* This predicate tests general unification.                   */
/* We call it general unification, because it cannot            */
/* be analysed at compile time. Therefore this kind of         */
/* unification cannot be compiled and, even in                 */
/* a compiled system, it must be handled at                    */
/* run time, exactly as by an interpreter.                     */
/* This is done by a general procedure for unification.        */
/* The name of the benchmark therefore does not                */
/* reflect that the unification is general, i.e. including     */
/* all Prolog types (e.g. it does not contain variables),      */
/* but it reflects the use of the procedure for general        */
/* unification as opposed to specific, compiled unification.   */

/* suggested value for N: 200 (interp), 500(comp)              */
/* results for Cprolog: N=200                                  */
/* Tloop=1.38 Tcomp=0.18 Tnet=1.20 Klips=0.17                  */

general_unification(X) :-
        nested_structure1(A),
        nested_structure2(B),
        T1 is cputime,
        do_general_unification(X,A,B),
        T2 is cputime,
        compens_loop(X),
        T3 is cputime,
        print_times(T1,T2,T3,X,1).

/* predicate to print the results of the benchmarking    */
```

```
print_times(T1,T2,T3,X,I) :-
        TT1 is T2 - T1,
        TT2 is T3 - T2,
        TT is TT1 - TT2,
        write('T overall loop:  '),write(TT1), nl,
        write('T compens loop:  '),write(TT2), nl,
        write('T benchmark:     '),write(TT),nl,
        write('KLips:           '),
        Li is I * X,
        Lips is Li / TT,
        KLips is Lips / 1000,
        write(KLips),nl,nl.

/* compensation loop, used to measure the time spent in the loop */
compens_loop(0).
compens_loop(X) :- Y is X - 1, compens_loop(Y).

/* list constructing loop */
do_construct_list(0).
do_construct_list(X) :- cl1(Z1,Z2,Z3), Y is X - 1,
                        do_construct_list(Y).

/* list matching loop */
do_match_list(0,Z).
do_match_list(X,Z) :- cl1(Z,Z,Z), Y is X - 1, do_match_list(Y,Z).

/* structure constructing loop */
do_construct_structure(0).
do_construct_structure(X) :- cs1(Z1,Z2,Z3), Y is X - 1,
                             do_construct_structure(Y).

/* structure matching loop */
do_match_structure(0,Z).
do_match_structure(X,Z) :- cs1(Z,Z,Z), Y is X - 1,
                           do_match_structure(Y,Z).

/* loop to match a nested structure */
do_match_nested_structure(0,Z).
do_match_nested_structure(X,Z) :-
                            nested_structure2(Z),
                            Y is X - 1,
                            do_match_nested_structure(Y,Z).

/* loop for general unification */
do_general_unification(0,A,B).
do_general_unification(X,A,B) :-
                            unify(A,B),
                            Y is X - 1,
                            do_general_unification(Y,A,B).

/* general unification */
unify(X,X).

/* complex structure as example for unification tests        */

/* the same structure is given twice, in order to make       */
/* sure that even implementations using structure sharing    */
/* execute the unification and do not just pass pointers    */

nested_structure1(
[a(     [a1([1,2,3],a),a2([4,5,6],b),a3([7,8,9],c)],
        [a4([0,1,2],d),a5([3,4,5],e),a6([6,7,8],f)],
        [a7([9,0,1],g),a8([2,3,4],h),a9([5,6,7],i)]),
 b(     [b1([1,2,3],a),b2([4,5,6],b),b3([7,8,9],c)],
        [b4([0,1,2],d),b5([3,4,5],e),b6([6,7,8],f)],
        [b7([9,0,1],g),b8([2,3,4],h),b9([5,6,7],i)]),
 c(     [c1([1,2,3],a),c2([4,5,6],b),c3([7,8,9],c)],
        [c4([0,1,2],d),c5([3,4,5],e),c6([6,7,8],f)],
        [c7([9,0,1],g),c8([2,3,4],h),c9([5,6,7],i)])]).

nested_structure2(
[a(     [a1([1,2,3],a),a2([4,5,6],b),a3([7,8,9],c)],
        [a4([0,1,2],d),a5([3,4,5],e),a6([6,7,8],f)],
        [a7([9,0,1],g),a8([2,3,4],h),a9([5,6,7],i)]),
 b(     [b1([1,2,3],a),b2([4,5,6],b),b3([7,8,9],c)],
        [b4([0,1,2],d),b5([3,4,5],e),b6([6,7,8],f)],
        [b7([9,0,1],g),b8([2,3,4],h),b9([5,6,7],i)]),
 c(     [c1([1,2,3],a),c2([4,5,6],b),c3([7,8,9],c)],
        [c4([0,1,2],d),c5([3,4,5],e),c6([6,7,8],f)],
        [c7([9,0,1],g),c8([2,3,4],h),c9([5,6,7],i)])]).

/* list of 100 elements used for match_list */
list100([a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
         a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
         a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
         a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a]).

/* structure of 100 elements used for match_structure     */
structure100(st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,st(a,
             st(a,nil)))))))))))))))))))))))))))))))))))
             )))))))))))))))))))))))))))))
             ))))))))))))))))))))))))))))))))))))))))).

/* predicates to test unification of lists */
cl1([a|X],[a|Y],[a|Z]) :- cl2(X,Y,Z).
cl2([a|X],[a|Y],[a|Z]) :- cl3(X,Y,Z).
        :
        :
cl99([a|X],[a|Y],[a|Z]) :- cl100(X,Y,Z).
cl100([a],[a],[a]).

/* predicates to test unification of structures */

cs1(st(a,X),st(a,Y),st(a,Z)) :- cs2(X,Y,Z).
cs2(st(a,X),st(a,Y),st(a,Z)) :- cs3(X,Y,Z).
        :
        :
cs99(st(a,X),st(a,Y),st(a,Z)) :- cs100(X,Y,Z).
cs100(st(a,nil),st(a,nil),st(a,nil)).

/* 1.6 Program to test dereferencing  (deref(N)).                   */

/* Program to benchmark the dereferencing speed.                    */
/* It constructs a list containing 500 variables which are          */
/* then bound together. Since different systems use different       */
/* strategies for binding variables on the global stack,            */
/* the whole is made for two lists                                  */
/* and the long variable chain is created only in one of them.      */
/* No compensation loop is therefore necessary.                     */
/* Suggested value for N is over 100;                               */
/* Results for Cprolog, N=200:                                      */
```

```
/* Tfirst loop=1.23 Tsecond loop=9.4 Tnet=8.16 Klips=0.29        */

deref(N):-
        make_list(500, L1, _),
        make_list(500, L2, Last),
        bind_forward(L1),
        bind_backward(L2),
        L2 = [a|_],
        T1 is cputime,
        ref(N, L1),
        T2 is cputime,
        ref(N, Last),
        T3 is cputime,
        print_times(T1,T2,T3,N,L2).


print_times(T1,T2,T3,X,I) :-        /* prints the results */
        TT1 is T2 - T1,
        TT2 is T3 - T2,
        abs_diff(TT1, TT2, TT),
        write('T first loop:    '),write(TT1), nl,
        write('T second loop:   '),write(TT2), nl,
        write('T net:           '),write(TT),nl,
        write('KLips:           '),
        Li is I * X,
        Lips is Li / TT,
        KLips is Lips / 1000,
        write(KLips),nl,nl.

abs_diff(X,Y,Z) :- X > Y, !, Z is X - Y.
abs_diff(X,Y,Z) :- Z is Y - X.

/*
/* Bind repeatively a cons cell to another one.
/*
ref(0, _) :- !.
ref(N, Cons) :-
        Cons = [a|_],
        Cons = [a|_],
        Cons = [a|_],
        Cons = [a|_],
        Cons = [a|_],
        Cons = [a|_],
        Cons = [a|_],
        Cons = [a|_],
        Cons = [a|_],
        Cons = [a|_],
        N1 is N - 1,
        ref(N1, Cons).

/*
/* Create a variable chain if in ?- equal(X, Y) the system binds
/* X to Y.

bind_forward([a]) :- !.
bind_forward([X, Y|T]) :-
        equal(X, Y),
        bind_forward([Y|T]).

/*
/* Create a variable chain if in ?- equal(X, Y) the system binds
/* Y to X.

bind_backward([X]) :- !.
bind_backward([X, Y|T]) :-
        bind_backward([Y|T]),
        equal(X, Y).

equal(X, X).

/*
/* Create a list containing variables and return the pointer to the
/* first and to the last cons cell.

make_list(1, L, L) :-
        L = [X],
        !.
make_list(N, [X|Rest], Last) :-
        N1 is N - 1,
        make_list(N1, Rest, Last).

/* 1.7. Program to test the cut operation.                        */

cuttest(N):-T1 is cputime,
        cutitl1(N), T2 is cputime,
        compens_loop(N), T3 is cputime,
        print_times(T1,T2,T3,N,300).

compens_loop(0).
compens_loop(X) :- Y is X - 1, compens_loop(Y).

/* cutitl1(N)    */
cutitl1(0).
cutitl1(N):- cuttl([100,100,100,100,100,100,100,100,100,100,
                    100,100,100,100,100,100,100,100,100,100,
                    100,100,100,100,100,100,100,100,100,100,
                    100,100,100,100,100,100,100,100,100,100,
                    100,100,100,100,100,100,100,100,100,100,
                    100,100,100,100,100,100,100,100,100,100,
                    100,100,100,100,100,100,100,100,100,100,
                    100,100,100,100,100,100,100,100,100,100,
                    100,100,100,100,100,100,100,100,100,100]),
             M is N-1, cutitl1(M).

cuttl([]).
cuttl([X|L]):-X=100, !, cuttl(L).
cuttl([X|L]):-X > 100, cuttl(L).

print_times(T1,T2,T3,X,I) :-        /* prints the results */
        TT1 is T2 - T1,
        TT2 is T3 - T2,
        TT is TT1 - TT2,
        write('T overall loop:  '),write(TT1), nl,
        write('T compens loop:  '),write(TT2), nl,
        write('T net:           '),write(TT),nl,
        write('KLips:           '),
        Li is I * X,
        Lips is Li / TT,
        KLips is Lips / 1000,
        write(KLips),nl,nl.
```

```
*********************************************
*** BENCHMARK PROGRAMS FOR PROLOG SYSTEMS   ***
***          (FINAL VERSION)               ***
*********************************************
```

Part 3 (of 3)

2. Small Prolog programs.

These benchmarks were run on a VAX 785 with 8 Meg of memory, under 4.2 BSD Unix. The interpreter was C-Prolog version 1.5.

This entire file (without mail/net headers) contains 584 lines.

Name	Call by	# of Inferences (one iteration)	KLips (C-Prolog)
fib	fibonacci(1).	4932	2.0
map	map(200).	68	1.3
mham	mham(1).	493824	1.7
mutest	mutest(1).	1366	2.3
quicksort	qs(10).	601	1.9
queens	qu(10).	684	1.7
query	query(1).	2294	0.9
sym_diff	differen(150).	71	1.5
diff_lists	diff(50).	608	2.1
nrev 10	nrev.	66	2.0
nrev 30	nrev.	496	2.5
nrev 50	nrev.	1326	2.5
nrev 100	nrev.	5151	2.5
nrev 150	nrev.	11476	2.5
nrev 200	nrev.	20301	2.5

```
/* Common functions...        */

print_times(T1,T2,T3,L) :-
        TT1 is T2 - T1,
        TT2 is T3 - T2,
        TT is TT1 - TT2,
        write('Net Time is: '), write(TT), nl,
        Lips is L / TT,
        Klips is Lips / 1000,
        write('KLips are: '), write(Klips), nl.

compens_loop(0).
compens_loop(X) :- Y is X - 1, compens_loop(Y).

el(X,[X|L]).
el(X,[Y|L]):-el(X,L).

list50([27,74,17,33,94,18,46,83,65,2,
       32,53,28,85,99,47,28,82,6,11,
       55,29,39,81,90,37,10,0,66,51,
        7,21,85,27,31,63,75,4,95,99,
       11,28,61,74,18,92,40,53,59,8]).

/* Fibonacci Series the slow way            */
/* fibonacci(1) will do...                  */

fibonacci(N) :- X is cputime,
          fib_loop(N),
          Now is cputime,
          compens_loop(N),
          M is cputime,
          Li is 4932 * N,
          print_times(X,Now,M,Li).


fib_loop(0).
fib_loop(X) :- top_fib(15,Z), Y is X - 1, fib_loop(Y).

top_fib(0,1).
top_fib(1,1).
top_fib(X,Y):-X1 is X-1,X2 is X-2,top_fib(X1,Y1),
              top_fib(X2,Y2),Y is Y1+Y2.

/* ---------------------------------------- */
/* Map colouring problem                    */
/*  map(200) is advised.                    */

map(N) :- X is cputime,
          map_loop(N),
          Now is cputime,
          compens_loop(N),
          M is cputime,
          Li is 68 * N,
          print_times(X,Now,M,Li).

map_loop(0).
map_loop(X) :- map_top, Y is X - 1, map_loop(Y).

map_top:-
el(X1,[b]),
el(X2,[r]),
el(X7,[g]),
el(X13,[w]),
el(X3,[b,r,g,w]),
not(X2=X3),
not(X3=X13),
el(X4,[b,r,g,w]),
not(X2=X4),
not(X7=X4),
not(X3=X4),
el(X5,[b,r,g,w]),
not(X13=X5),
not(X3=X5),
not(X4=X5),
el(X6,[b,r,g,w]),
not(X13=X6),
not(X5=X6),
el(X8,[b,r,g,w]),
not(X7=X8),
not(X13=X8),
el(X9,[b,r,g,w]),
not(X13=X9),
not(X4=X9),
not(X8=X9),
el(X10,[b,r,g,w]),
```

```
not(X4=X10),
not(X5=X10),
not(X6=X10),
not(X9=X10),
el(X11,[b,r,g,w]),
not(X11=X13),
not(X11=X10),
not(X11=X6),
el(X12,[b,r,g,w]),
not(X12=X13),
not(X12=X11),
not(X12=X9),
display([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13]),nl.

map_top.

/* ---------------------------------------------- */
/*  Hamiltonian Graphs...                         */
/*  Extremely long (nearly half a million LI's !) */
/*  Only 1 advised !                              */

mham(N) :- X is cputime,
           mham_loop(N),
           Now is cputime,
           compens_loop(N),
           M is cputime,
           Li is 493824 * N,
           print_times(X,Now,M,Li).

mham_loop(0).
mham_loop(X) :- mham_top, Y is X - 1, mham_loop(Y).

mham_top:-
        cycle_ham([a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t],X),
        fail.
mham_top.

cycle_ham([X|Y],[X,T|L]):-
        chain_ham([X|Y],[],[T|L]),
        edge(T,X).

chain_ham([X],L,[X|L]).
chain_ham([X|Y],K,L):-
        delete(Z,Y,T),
        edge(X,Z),
        chain_ham([Z|T],[X|K],L).

delete(X,[X|Y],Y).
delete(X,[U|Y],[U|Z]):-
        delete(X,Y,Z).

edge(X,Y):-
        connect(X,L),
        el(Y,L).

connect(0,[1,2,3,4,5,6,7,8,9]).
connect(1,[0,2,3,4,5,6,7,8,9]).
    :
    :
connect(8,[0,1,2,3,4,5,6,7,9]).
connect(9,[0,1,2,3,4,5,6,7,8]).

connect(a,[b,j,k]).
connect(b,[a,c,p]).
connect(c,[b,d,l]).
connect(d,[c,e,q]).
connect(e,[d,f,m]).
    :
    :
connect(p,[b,q,t]).
connect(q,[p,r,d]).
connect(r,[q,s,f]).
connect(s,[r,t,h]).
connect(t,[p,s,j]).

/* ---------------------------------------------- */
/*  Hofstader's mu math (mutest) proving muiiu    */
/*  from Godel Escher Bach                        */
mutest(N) :- X is cputime,
           mu_loop(N),
           Now is cputime,
           compens_loop(N),
           M is cputime,
           Li is 1366 * N,
           print_times(X,Now,M,Li).

mu_loop(0).
mu_loop(X) :- mu_top, Y is X - 1, mu_loop(Y).

mu_top:- theorem(5,[m,u,i,i,u]).

rules(S, R) :- rule3(S,R).
rules(S, R) :- rule4(S,R).
rules(S, R) :- rule1(S,R).
rules(S, R) :- rule2(S,R).

rule1(S,R) :-
        append(X, [i], S),
        append(X, [i,u], R).

rule2([m | T], [m | R]) :- append(T, T, R).

rule3([], -) :- fail.
rule3(R, T) :-
        append([i,i,i], S, R),
        append([u], S, T).
rule3([H | T], [H | R]) :- rule3(T, R).

rule4([], -) :- fail.
rule4(R, T) :- append([u, u], T, R).
rule4([H | T], [H | R]) :- rule4(T, R).

theorem(Depth, [m, i]).
theorem(Depth, []) :- fail.

theorem(Depth, R) :-
        Depth > 0,
        D is Depth - 1,
        theorem(D, S),
        rules(S, R).

append([], X, X).
append([A | B], X, [A | Bl]) :-
        !,
        append(B, X, Bl).

/* ---------------------------------  */
/*  Quicksort of 50 element list      */
/*                                    */

qs(N) :-  list50(L),
          X is cputime,
```

```
        qs_loop(N,L),
        Now is cputime,
        compens_loop(N),
        M is cputime,
        Li is 601 * N,
        print_times(X,Now,M,Li).

qs_loop(0,_).
qs_loop(X,L) :- qsort(L,Z,[]), Y is X - 1,qs_loop(Y,L).

qsort([X|L],R,R0) :-
      partition(L,X,L1,L2),
      qsort(L2,R1,R0),
      qsort(L1,R,[X|R1]).
qsort([],R,R).

partition([X|L],Y,[X|L1],L2) :- X =< Y,!,
      partition(L,Y,L1,L2).
partition([X|L],Y,L1,[X|L2]) :-
      partition(L,Y,L1,L2).
partition([],_,[],[]).

/* ---------------------------------- */
/*  Queens on a chess board problem...  */
/*  Only two solution on a 4x4 board... */
/*  about 5 - 10 is advised for N.      */
qu(N) :- X is cputime,
         qu_nloop(N),
         Now is cputime,
         compens_loop(N),
         M is cputime,
         Li is 684 * N,
         print_times(X,Now,M,Li).

qu_nloop(0).
qu_nloop(X) :- not(qu_top), Y is X - 1, qu_nloop(Y).

qu_top :-  run(4,X), fail.

size(4).
snint(1).
snint(2).
snint(3).
snint(4).

run(Size, Soln) :- get_solutions(Size, Soln).

get_solutions(Board_size, Soln) :- solve(Board_size, [], Soln).

/*  newsquare generates legal positions for next queen     */

newsquare([], square(1, X)) :- snint(X).
newsquare([square(I, J) | Rest], square(X, Y)) :-
       X is I + 1,
       snint(Y),
       not(threatened(I, J, X, Y)),
       safe(X, Y, Rest).

/*   safe checks whether square(X, Y) is threatened by any */
/*   existing queens                                       */

safe(X, Y, []).
safe(X, Y, [square(I, J)   L]) :-
       not(threatened(I, J, X, Y)),
       safe(X, Y, L).
```

```
/*    threatened checks whether squares (I, J) and (X, Y) */
/*    threaten each other                                 */

threatened(I, J, X, Y) :-
       (I = X),
       !.
threatened(I, J, X, Y) :-
       (J = Y),
       !.
threatened(I, J, X, Y) :-
       (U is I - J),
       (V is X - Y),
       (U = V),
       !.
threatened(I, J, X, Y) :-
       (U is I + J),
       (V is X + Y),
       (U = V),
       !.

/* solve accumulates the positions of occupied squares */

solve(Bs, [square(Bs, Y) | L], [square(Bs, Y) | L]) :- size(Bs).
solve(Board_size, Initial, Final) :-
       newsquare(Initial, Next),
       solve(Board_size, [Next | Initial], Final).

/* ---------------------------------- */
/* Query does simple database queries.  */
/*                                      */

query(N) :- X is cputime,
         que_nloop(N),
         Now is cputime,
         compens_loop(N),
         M is cputime,
         Li is 2294 * N,
         print_times(X,Now,M,Li).

que_nloop(0).
que_nloop(X) :- not(que_top), Y is X - 1, que_nloop(Y).

que_top:-
        que(X),
        fail.

que([C1,D1,C2,D2]) :-
      density(C1,D1),
      density(C2,D2),
      D1>D2,
      20*D1<21*D2.

density(C,D) :-
      pop(C,P),
      area(C,A),
      D is (P*100)/A.

pop(china,8250).
pop(india,5863).
pop(ussr,2521).
pop(usa,2119).
pop(indonesia,1276).
pop(japan,1097).
pop(brazil,1042).
pop(bangladesh,750).
pop(pakistan,682).
```

```
pop(w_germany,620).
pop(nigeria,613).
pop(mexico,581).
pop(uk,559).
pop(italy,554).
pop(france,525).
pop(philippines,415).
pop(thailand,410).
pop(turkey,383).
pop(egypt,364).
pop(spain,352).
pop(poland,337).
pop(s_korea,335).
pop(iran,320).
pop(ethiopia,272).
pop(argentina,251).

area(china,3380).
area(india,1139).
area(ussr,8708).
area(usa,3609).
area(indonesia,570).
area(japan,148).
area(brazil,3288).
area(bangladesh,55).
area(pakistan,311).
area(w_germany,96).
area(nigeria,373).
area(mexico,764).
area(uk,86).
area(italy,116).
area(france,213).
area(philippines,90).
area(thailand,200).
area(turkey,296).
area(egypt,386).
area(spain,190).
area(poland,121).
area(s_korea,37).
area(iran,628).
area(ethiopia,350).
area(argentina,1080).

/* ------------------------------------------------*/
/*       differen (times10,divide10,log10,ops8)    */
/*       These 4 examples are from Warren's thesis */
/*       differen(150) will do.                    */

differen(N) :- X is cputime,
          differenloop(N),
          Now is cputime,
          compens_loop(N),
          M is cputime,
          Li is 71 * N,
          print_times(X,Now,M,Li).

differenloop(0).
differenloop(X) :- differen_top, Y is X - 1, differenloop(Y).

differen_top:-
        times10(I1),
        d(I1,x,D1),
        divide10(I2),
        d(I2,x,D2),
        log10(I3),
        d(I3,x,D3),
        ops8(I4),
        d(I4,x,D4).

d(U+V,X,DU+DV) :- !, d(U,X,DU), d(V,X,DV).
d(U-V,X,DU-DV) :- !, d(U,X,DU), d(V,X,DV).
d(U*V,X,DU*V+U*DV) :- !, d(U,X,DU), d(V,X,DV).
d(U/V,X,(DU*V-U*DV)/(^(V,2))) :- !, d(U,X,DU), d(V,X,DV).
d(^(U,N),X,DU*N*(^(U,N1))) :- !, integer(N), N1 is N - 1, d(U,X,DU).
d(-U,X,-DU) :- !, d(U,X,DU).
d(exp(U),X,exp(U)*DU) :- !, d(U,X,DU).
d(log(U),X,DU/U) :- !, d(U,X,DU).
d(X,X,1).
d(C,X,0).

times10( (((((((((x*x)*x)*x)*x)*x)*x)*x)*x)*x)*x ).
divide10( (((((((((x/x)/x)/x)/x)/x)/x)/x)/x)/x)/x ).
log10( log(log(log(log(log(log(log(log(log(log(x)))))))))) ).
ops8( (x+1)*((^(x,2)+2)*(^(x,3)+3)) ).

/* ------------------------------------------------ */
/*  Difference Lists                                */
/*       quicksort on 50 items (difference lists)   */

diff(N) :- list50(L),
          X is cputime,
          difflistloop(N,L),
          Now is cputime,
          compens_loop(N),
          M is cputime,
          Li is 608 * N,
          print_times(X,Now,M,Li).

difflistloop(0,_).
difflistloop(X,L) :- qdsort(L,Z), Y is X - 1, difflistloop(Y,L).

qdsort([X|L],R-R0) :-
        dpartition(L,X,L1,L2),
        qdsort(L1,R-[X|R1]),
        qdsort(L2,R1-R0).
qdsort([],R0-R0).

dpartition([X|L],Y,[X|L1],L2) :-
        X<Y, !,
        dpartition(L,Y,L1,L2).
dpartition([X|L],Y,L1,[X|L2]) :-
        dpartition(L,Y,L1,L2).
dpartition([],_,[],[]).

/* ------------------------------------------------ */
/*  Naive reverse for variable length lists...     */
/*  try with 10, 30, 50, 100, 150, 200.            */

nrev:- write('list length: '),
        read(X),
        conslist(X, List),
        T1 is cputime,
        nreverse(List, _),
        T2 is cputime,
        T is T2 - T1,
        I is (X*(X+3))/2 + 1,
        LIPS is I/T,
        write('LIPS= '),
        write(LIPS).

nreverse([], []).
nreverse([X|L0],L) :- nreverse(L0, L1),
```

```
    concatenate(L1, [X], L).

concatenate([], L, L).
concatenate([X|L1], L2, [X|L3]) :- concatenate(L1, L2, L3).

conslist(0, []) :- !.
conslist(N, [N|L]) :-
        N1 is N-1,
        conslist(N1, L).

/* 3. Real Prolog programs.

This section is empty for now. ....

*/
```

```
% Main program to do branch and bound NAND gate designs.
% Optimized for 2-input NAND gates and 3 input variables.
% By A. Despain, Feb 84. Univ. California, Berkeley
% In this case, design a 2-1 MUX (ckt2)

main :- set_bound(32000), update_circuit([],0), r(3, [0,0,1,1,0,1,0,1]).

run(Depth, Table, Circuit, Cost, Depth) :- search(Depth, Table),
                                           circuit(Circuit),
                                           Circuit\==[],
                                           cost_bound(Cost),!.
run(Depth, Table, Circuit, Cost, Delay) :- D is Depth + 1,
    run(D, Table, Circuit, Cost, Delay),!.

search(Depth, Table) :- t(Depth, Circuit, Table, 0, Cost_out),
                        update_circuit(Circuit,Cost_out),
                        update_bound(Cost_out).

% Input signals are free and terminate the search.
t(_, 0 , [0,1,0,1,0,1,0,1],C,C).
t(_, 1 , [0,0,1,1,0,0,1,1],C,C).
t(_, 2 , [0,0,0,0,1,1,1,1],C,C).
t(_,i0 , [1,0,1,0,1,0,1,0],C,C).
t(_,i1 , [1,1,0,0,1,1,0,0],C,C).
t(_,i2 , [1,1,1,1,0,0,0,0],C,C).

% Inverters are free in this model.
t(Depth, [i,Z], Table, Cin, Cout) :- Depth > 0,
                                     D is Depth -1,
                                     sint(Table, Itable),
                                     t(D, Z, Itable, Cin, Cout).

% Main NAND gate clause.
t(Depth, [X,Y,Z], Table, Cin, Cout) :- Depth > 0,
                                     D is Depth -1,
                                     update_cost(Cin,1,C2),
                                     ngate(Table, A, B),
                                     t(D,Y,A,C2,C3),
                                     t(D,Z,B,C3,Cout).

% Inverter signal transformation.
%sint([H1,..T1],[H2,..T2]) :- inv(H1, H2), sint(T1, T2).
sint([],[]).
sint([X,.T1],[_,..T2]) :- var(X), sint(T1, T2),!.
sint([0,..T1],[1,..T2]) :- sint(T1, T2).
sint([1,..T1],[0,..T2]) :- sint(T1, T2).

% Optimized gate signal transformation.
ngate([], [], 1).
tgate([], [], 1).
ngate([X,..T0], [_,..T1], [_,..T2]) :- var(X), !, ngate(T0, T1, T2).
ngate([X,..T0], [1,..T1], [1,..T2]) :- X==0, ngate(T0, T1, T2).
ngate([X,..T0], [_,..T1], [0,..T2]) :- X==1, tgate(T0, T1, T2).
tgate([X,..T0], [_,..T1], [_,..T2]) :- var(X), !, tgate(T0, T1, T2).
tgate([X,..T0], [1,..T1], [1,..T2]) :- X==0, tgate(T0, T1, T2).
tgate([X,..T0], [_,..T1], [0,..T2]) :- X==1, tgate(T0, T1, T2).
tgate([X,..T0], [0,..T1], [_,..T2]) :- X==1, tgate(T0, T1, T2).

r(Depth,Table) :- run(3, Table, L, C, D),
                  Depth =< D,
                  nl, write([minimum,cost,circuit,of,the,shortest,delay]),
                  nl, write([  circuit,=,L]),
                  nl, write([  cost,is,C,gates]),
                  nl, write([  delay,is,D,gate,times]),nl,!.
r(Depth,Table) :- run(Depth, Table, L, C, D),
                  nl, write([lowest,cost,circuit,for,a,given,delay]),
                  nl, write([  circuit,=,L]),
                  nl, write([  cost,is,C,gates]),
                  nl, write([  delay,is,D,gate,times]),nl.

%Utility procedures

min(X,Y,X) :- X < Y , ! .
min(X,Y,Y).

update_cost(Cost_in, Cost, Cost_out) :- Cost_out is Cost_in + Cost,
                                        cost_bound(B),
                                        Cost_out < B, ! .

cost_bound(32000).

set_bound(X) :- retract([cost_bound(_)]),
                assert([cost_bound(X)]), ! .

update_bound(X) :- retract([cost_bound(Y)]),
                   min(X,Y,Z),
                   assert([cost_bound(Z)]), ! .

update_circuit(Circuit,Cost) :- cost_bound(X),
                                Cost < X ,
                                retract([circuit(_)]),
                                assert([circuit(Circuit)]),!.
update_circuit(Circuit,Cost).

circuit([]).

-------------------------------------------------------------------------------
%       concat (con1, con6)
%       These two tests are simple examples of the concat predicate
%       con1 is determinate, con6 is non-determinate getting all 6 answers

main1 :- concat([a,b,c],[d,e],X),  % con1
         write(X),nl.
main6 :- concat(X,Y,[a,b,c,d,e]),  % con6
         write(X),nl,
         write(Y),nl,nl,
         fail.

concat([],L,L).
concat([X|L1],L2,[X|L3]) :- concat(L1,L2,L3).
-------------------------------------------------------------------------------
%       differen (times10,divide10,log10,ops8)
%       These 4 examples are from Warren's thesis

main :-
         times10(I1),
         d(I1,x,D1),
         write(D1), nl,
         divide10(I2),
         d(I2,x,D2),
         write(D2), nl,
         log10(I3),
         d(I3,x,D3),
         write(D3), nl,
         ops8(I4),
         d(I4,x,D4),
         write(D4), nl.

d(U+V,X,DU+DV) :- !, d(U,X,DU), d(V,X,DV).
d(U-V,X,DU-DV) :- !, d(U,X,DU), d(V,X,DV).
d(U*V,X,DU*V+U*DV) :- !, d(U,X,DU), d(V,X,DV).
d(U/V,X,(DU*V-U*DV)/(^(V,2))) :- !, d(U,X,DU), d(V,X,DV).
```

```
d(^(U,N),X,DU*N*(^(U,N1))) :- !, integer(N), N1 is N - 1, d(U,X,DU).
d(-U,X,-DU) :- !, d(U,X,DU).
d(exp(U),X,exp(U)*DU) :- !, d(U,X,DU).
d(log(U),X,DU/U) :- !, d(U,X,DU).
d(X,X,1).
d(C,X,0).

times10( (((((((((X*X)*X)*X)*X)*X)*X)*X)*X)*X)*X ).
divide10( ((((((((X/X)/X)/X)/X)/X)/X)/X)/X)/X ).
log10( log(log(log(log(log(log(log(log(log(log(X)))))))))) ).
ops8( (X+1)*((^(X,2)+2)*(^(X,3)+3)) ).
```



```
%       towers of hanoi ( hanoi ) for 8 disks

main :- hanoi(4).

hanoi(N) :- move(N,left,center,right).

move(0,_,_,_) :- !.
move(N,A,B,C) :- M is N-1, move(M,A,C,B), inform(A,B), move(M,C,B,A).

inform(A,B) :- write([move,disk,from,A,to,B]), nl, fail.
inform(_,_).
```

```
%  Hofstader's mu math (mutest) proving muiiu
%       from Godel Escher Bach

main :- theorem(5,[m,u,i,i,u]).

rules(S, R) :- rule3(S,R).
rules(S, R) :- rule4(S,R).
rules(S, R) :- rule1(S,R).
rules(S, R) :- rule2(S,R).

rule1(S,R) :-
        append(X, [i], S),
        append(X, [i,u], R).

rule2([m | T], [m | R]) :- append(T, T, R).

rule3([], _) :- fail.
rule3(R, T) :-
        append([i,i,i], S, R),
        append([u], S, T).
rule3([H | T], [H | R]) :- rule3(T, R).

rule4([], _) :- fail.
rule4(R, T) :- append([u, u], T, R).
rule4([H | T], [H | R]) :- rule4(T, R).

theorem(Depth, [m, i]).
theorem(Depth, []) :- fail.

theorem(Depth, R) :-
        Depth > 0,
        D is Depth - 1,
        theorem(D, S),
        rules(S, R).

append([], X, X).
append([A | B], X, [A | Bl]) :-
        !,
        append(B, X, Bl).
```

```
%       naive reverse (nrev1)
%       from Warren's thesis

main :-
        list30(L),
        nreverse(L,X),
        write(X), nl.

nreverse([X|L0],L) :- nreverse(L0,L1), concatenate(L1,[X],L).
nreverse([],[]).

concatenate([X|L1],L2,[X|L3]) :- concatenate(L1,L2,L3).
concatenate([],L,L).

list30([1,2,3,4,5,6,7,8,9,10,11,12,
        13,14,15,16,17,18,19,20,21,
        22,23,24,25,26,27,28,29,30]).
```

```
%       serialize (palin25)
%       from Warren's thesis

main :-
        palin25(P),
        serialize(P,X),
        write(X),nl.

serialize(L,R) :-
        pairlists(L,R,A),
        arrange(A,T),
        numbered(T,1,N).

pairlists([X|L], [Y|R], [pair(X,Y)|A]) :- pairlists(L,R,A).
pairlists([], [], []).

arrange([X|L], tree(T1, X, T2)) :-
        split(L, X, L1, L2),
        arrange(L1, T1),
        arrange(L2, T2).
arrange([], void).

split([X|L], X, L1, L2) :- !, split(L, X, L1, L2).
split([X|L], Y, [X|L1], L2) :- before(X,Y), !, split(L,Y,L1,L2).
split([X|L], Y, L1, [X|L2]) :- before(Y,X), !, split(L,Y,L1,L2).
split([], _, [], []).

before(pair(X1,Y1), pair(X2,Y2)) :- X1 < X2.

numbered(tree(T1, pair(X,N1), T2), N0, N) :-
        numbered(T1, N0, N1),
        is(N2,N1,+,1),
        numbered(T2,N2,N).
numbered(_,N,N).

palin25("ABLE WAS I ERE I SAW ELBA").
```

```
% The sieve of Eratosthenes, from Clocksin & Mellish (pri2)
%       finding the prime numbers up to 98.

main :- primes(98, X), write(X), nl.

primes(Limit, Ps) :- integers(2, Limit, Is), sift(Is, Ps).

integers(Low, High, [Low | Rest]) :-
        Low =< High, !,
        M is Low+1,
        integers(M, High, Rest).
integers(_,_,[]).

sift([],[]).
sift([I | Is], [I | Ps]) :- remove(I,Is,New), sift(New, Ps).

remove(P,[],[]).
remove(P,[I | Is], [I | Nis]) :- not(0 is I mod P), !, remove(P,Is,Nis).
remove(P,[I | Is], Nis) :- 0 is I mod P, !, remove(P,Is,Nis).
```

```
%       quicksort (qs4) on 50 items
%       from Warren's thesis

main :-
        list50(L),
        qsort(L,X,[]),
        write(X), nl.

qsort([X L],R,R0) :-
        partition(L,X,L1,L2),
        qsort(L2,R1,R0),
        qsort(L1,R,[X|R1]).
qsort([],R,R).

partition([X|L],Y,[X|L1],L2) :-
        X<Y, !,
        partition(L,Y,L1,L2).
partition([X|L],Y,L1,[X|L2]) :-
        partition(L,Y,L1,L2).
partition([],_,[],[]).

list50([27,74,17,33,94,18,46,83,65,2,
        32,53,28,85,99,47,28,82,6,11,
        55,29,39,81,90,37,10,0,66,51,
        7,21,85,27,31,63,75,4,95,99,
        11,28,61,74,18,92,40,53,59,8]).
```

```
%       the queens on a chessboard problem (queens) for 4x4 board

main :-  run(4,X), fail.

size(4).
int(1).
int(2).
int(3).
int(4).

run(Size, Soln) :- get_solutions(Size, Soln), inform(Soln).

get_solutions(Board_size, Soln) :- solve(Board_size, [], Soln).

%       newsquare generates legal positions for next queen

newsquare([], square(1, X)) :- int(X).
newsquare([square(I, J) | Rest], square(X, Y)) :-
        X is I + 1,
        int(Y),
        not(threatened(I, J, X, Y)),
        safe(X, Y, Rest).

%       safe checks whether square(X, Y) is threatened by any
%       existing queens

safe(X, Y, []).
safe(X, Y, [square(I, J) | L]) :-
        not(threatened(I, J, X, Y)),
        safe(X, Y, L).

%       threatened checks whether squares (I, J) and (X, Y)
%       threaten each other

threatened(I, J, X, Y) :-
        (I = X),
        !.
threatened(I, J, X, Y) :-
        (J = Y),
        !.
threatened(I, J, X, Y) :-
        (U is I - J),
        (V is X - Y),
        (U = V),
        !.
threatened(I, J, X, Y) :-
        (U is I + J),
        (V is X + Y),
        (U = V),
        !.

%       solve accumulates the positions of occupied squares

solve(Bs, [square(Bs, Y) | L], [square(Bs, Y) | L]) :- size(Bs).
solve(Board_size, Initial, Final) :-
        newsquare(Initial, Next),
        solve(Board_size, [Next | Initial], Final).

inform([]) :- nl,nl.
inform([M | L]) :- write(M), nl, inform(L).

% this is compiled code for not.

procedure not

        try_me_else     N1
        allocate
        escape          call,0
        cut
        fail

N1:     trust_me_else   fail
        proceed
```

```
%       query
%       from Warren's thesis

main :-
        query(X),
        write(X), nl,
        fail.

query([C1,D1,C2,D2]) :-
        density(C1,D1),
        density(C2,D2),
        D1 > D2,
        T1 is 20*D1,
        T2 is 21*D2,
        T1 < T2.

density(C,D) :- pop(C,P), area(C,A), D is (P*100)/A.

pop(china,      8250).   area(china,      3380).
pop(india,      5863).   area(india,      1139).
pop(ussr,       2521).   area(ussr,       8708).
pop(usa,        2119).   area(usa,        3609).
pop(indonesia,  1276).   area(indonesia,  570).
pop(japan,      1097).   area(japan,      148).
pop(brazil,     1042).   area(brazil,     3288).
pop(bangladesh, 750).    area(bangladesh,55).
pop(pakistan,   682).    area(pakistan,   311).
pop(w_germany,  620).    area(w_germany,  96).
pop(nigeria,    613).    area(nigeria,    373).
pop(mexico,     581).    area(mexico,     764).
pop(uk,         559).    area(uk,         86).
pop(italy,      554).    area(italy,      116).
pop(france,     525).    area(france,     213).
pop(phillipines,415).    area(phillipines,90).
pop(thailand,   410).    area(thailand,   200).
pop(turkey,     383).    area(turkey,     296).
pop(egypt,      364).    area(egypt,      386).
pop(spain,      352).    area(spain,      190).
pop(poland,     337).    area(poland,     121).
pop(s_korea,    335).    area(s_korea,    37).
pop(iran,       320).    area(iran,       628).
pop(ethiopia,   272).    area(ethiopia,   350).
pop(argentina,  251).    area(argentina,  1080).
```