UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2003

BEng Honours Degree in Computing Part III
MSc in Computing Science
PhD
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C327

THE PRACTICE OF LOGIC PROGRAMMING

Thursday 8 May 2003, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1a Write a program in Prolog that will find all the 3 x 3 magic squares (comprising the digits 1 to 9, with the sum of each row, column and principal diagonal the same) and print them out one after another, without user intervention:

```
 -----
|8 3 4|
|1 5 9|
|6 7 2|
 -----
 -----
|8 1 6|
|3 5 7|
|4 9 2|
 -----
...
 -----
|2 7 6|
|9 5 1|
|4 3 8|
 -----
```

b i) Write a program in Prolog, without using the cut, which will find the integer given by (all of) the following clues:

- If it is a multiple of 2 then it lies between 50 and 59 (inclusive).
- If it is not a multiple of 3 then it lies between 60 and 69 (inclusive).
- If it is not a multiple of 4 then it lies between 70 and 79 (inclusive).

ii) How could you modify the program in b(i), utilising cuts where they may lead to improved efficiency, but leaving the logic unaltered? There is no need to rewrite those clauses unchanged from b(i).

Part a and the subparts of b carry, respectively, 35%, 40%, 25% of the marks.

2a Every rational number $\frac{a}{b}$, where a and b are positive integers, can be written as finite continued fractions of the form:

$$q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{\cdots\cdots + \cfrac{1}{q_{n-1} + \cfrac{1}{q_n}}}}}}$$

where $q_1, q_2, \cdots, q_n$ are positive integers.

To represent $\frac{693}{147}$, say, as a continued fraction, you can proceed as follows:

$$\begin{aligned} 693 &= 4 \times 147 + 105 \\ 147 &= 1 \times 105 + 42 \\ 105 &= 2 \times 42 + 21 \\ 42 &= 2 \times 21 + 0 \end{aligned}$$

and so: $\frac{693}{147} = 4 + \cfrac{1}{1 + \cfrac{1}{2 + \frac{1}{2}}}$

i) Define a predicate `continued(A, B, C)` in Prolog, that can be used to find the continued fraction list C of integers in a representation of $\frac{A}{B}$, where A and B are given positive integers. For example, the query

```
?- continued(693, 147, C).
```

should return: `C = [4, 1, 2, 2].`

NOTE: You may use the built-in Prolog integer division and modulus infix operators // and mod.

ii) Define a predicate `continued1(A, B, C)` in Prolog that can be used to find a rational number $\frac{A}{B}$ represented by a given continued fraction list C of integers. Extra credit will be given, if `continued1` employs a tail-recursive auxiliary predicate.

b. Ancient Egyptians normally represented positive rational numbers, less than 1, as sums of distinct reciprocals of integers greater than 1. For example:

$\frac{5}{6}$ was represented by $\frac{1}{2}+\frac{1}{3}$

$\frac{18}{23}$ was represented by $\frac{1}{2}+\frac{1}{4}+\frac{1}{31}+\frac{1}{2852}$

The simplest algorithm for finding an Egyptian fraction representation of a rational $\frac{a}{b}$, less than 1, is to find the largest reciprocal $\frac{1}{n}$ that is less than or equal to it, to subtract $\frac{1}{n}$ from $\frac{a}{b}$, and then to repeat the process until nothing remains.

i) Define a predicate `biggest_recip(A, B, N)` in Prolog, meaning that $\frac{1}{N}$ is the greatest reciprocal, less than or equal to given positive rational number $\frac{A}{B}$, less than 1.

ii) Define a predicate `egyptian(A, B, E)` in Prolog, meaning that E is a list of distinct positive integers, greater than 1, whose reciprocals sum to given positive rational number $\frac{A}{B}$, less than 1.

NOTE. Because of possible floating point errors, it is unsafe to test whether two rationals are equal by first converting them to floating point numbers using ordinary division.

The four subparts each carry 25% of the marks.

3a Write the "demo" code, in Prolog, for a simple interpreter that handles negation as failure, conjunctions of subgoals and matching clauses in the same manner as Prolog. You may ignore system primitives, such as arithmetic goals, findall and cut. Your interpreter need neither produce explanations nor query the user.

b Assume the following knowledge base:

```
:- dynamic celeb/1.
:- dynamic player/1.
:- dynamic wife/2.

celeb(X) :-
     player(X).
celeb(X) :-
     \+ player(X),
     wife(X, Y),
     player(Y).

player(becks).

wife(posh,becks).
```

and the query

```
?- celeb(posh).
```

i) Draw an OR-tree showing how Prolog executes the above query.

ii) Extend the interpreter from part a, so that it produces a trace of its successful actions in proving goals. For example, the above query should produce a trace similar to:

```
goal: \+ player(posh) SUCCEEDS
goal: wife(posh,becks) SUCCEEDS
goal: player(becks) SUCCEEDS
goal: celeb(posh) SUCCEEDS
```

iii) Extend the interpreter from part a, so that it produces a full trace of its actions, matching goals with the heads of rules, indicating both success and failure in proving goals, as well as its strategy in dealing with negation. For example, the above query should produce a trace similar to:

```
trying goal: celeb(posh)
found matching rule: celeb(posh) :- player(posh).
trying goal: player(posh)
goal: player(posh) FAILS
found matching rule: celeb(posh) :- \+player(posh),wife(posh,_656),player(_656).
trying negated goal: \+ player(posh)
trying goal: player(posh)
goal: player(posh) FAILS
negated goal: \+ player(posh) SUCCEEDS
trying goal: wife(posh,_656)
found matching fact: wife(posh,becks).
goal: wife(posh,becks) SUCCEEDS
trying goal: player(becks)
found matching fact: player(becks).
goal: player(becks) SUCCEEDS
goal: celeb(posh) SUCCEEDS
```

The four parts and subparts carry, respectively, 20%, 20%, 20%, 40% of the marks.

4a What is the essential difference in the behaviour of Prolog and CLP(FD) programs designed to find solutions to problems?

b Write a program in CLP(FD) that can solve the following problem:

> *Four singers, one each from London, Manchester, Newcastle and Oxford gave a concert. One was dressed in black, one in blue, one in brown and one in burgundy. Their names were Adams, Baker, Clifford and Dennison.*
>
> *The Londoner and Clifford sang the duet from the Pearl Fishers. The singers in brown and burgundy sang the quartet from Rigoletto together with Baker and the singer from Newcastle.*
>
> *Dennison and the singer from Oxford really admired the singer in brown but were disappointed with the Manchester singer in black.*
>
> *Where did Adams, Baker, Clifford and Dennison come from and what was each wearing?*

Your program should produce its output in the following form:

```
Adams - London - Brown
Baker - Manchester - Black
Clifford - Oxford - Burgundy
Dennison - Newcastle - Blue
```

c Give a careful explanation of the purpose and behaviour of each section of the code in part b.

The three parts carry, respectively, 20%, 50%, 30% of the marks.