UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2002

BEng Honours Degree in Computing Part III
MSc in Computing Science
BSc Honours Degree in Mathematics and Computer Science Part III
MSci Honours Degree in Mathematics and Computer Science Part III
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
This paper is also taken for the relevant examinations for the Associateship of the Royal College of Science

PAPER C327

THE PRACTICE OF LOGIC PROGRAMMING

Thursday 25 April 2002, 14:00
Duration: 120 minutes

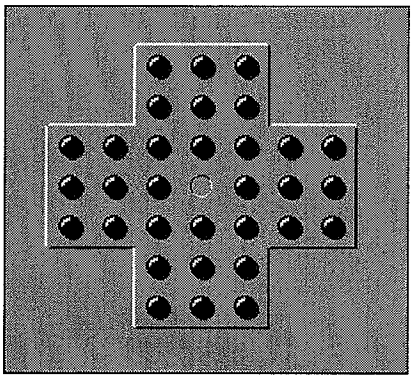
Answer THREE questions

Paper contains 4 questions
Calculators not required

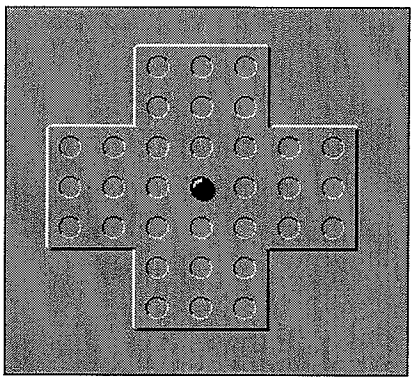
1a i) Give Prolog clauses for `remove(x,y,z)` where y is a list of distinct numbers, including x, and z is the list resulting from removing x from y.

ii) Give Prolog clauses for `insert(x,y,z)` where y is a list of distinct numbers in ascending order, not including number x, and z is the list resulting from inserting x into y, in the right place.

b *Peg Solitaire* is a game that consists of a board with 33 holes arranged in the pattern given in the pictures below. At the start, every hole except the centre is filled with a peg. The player can move any peg by jumping over an adjacent peg into an empty hole. Pegs that are jumped over are removed, just as in draughts. Vertical and horizontal jumps are allowed, but diagonal jumps are not. The goal is to reach the position, through a sequence of jumps, where there is only one peg left - in the centre hole.



Start



Finish

You can think of the board as being formed from a 7 by 7 grid in which sixteen cells at the corners are out-of-bound. So a natural representation of any state of the board is an ordered list of distinct integers, each between 0 and 48, giving the current positions of the pegs.

i) Represent in Prolog the start and goal states, and express the fact that the sixteen corner cells are out-of-bound.

ii) Give a Prolog procedure for `move(P, D, S1, S2)`, meaning that moving peg P in direction D (*left*, *right*, *down* or *up*) is a valid move in state S1, resulting in state S2.

iii) Write a Prolog program for solve(M) where M is a list of moves that solves the puzzle.

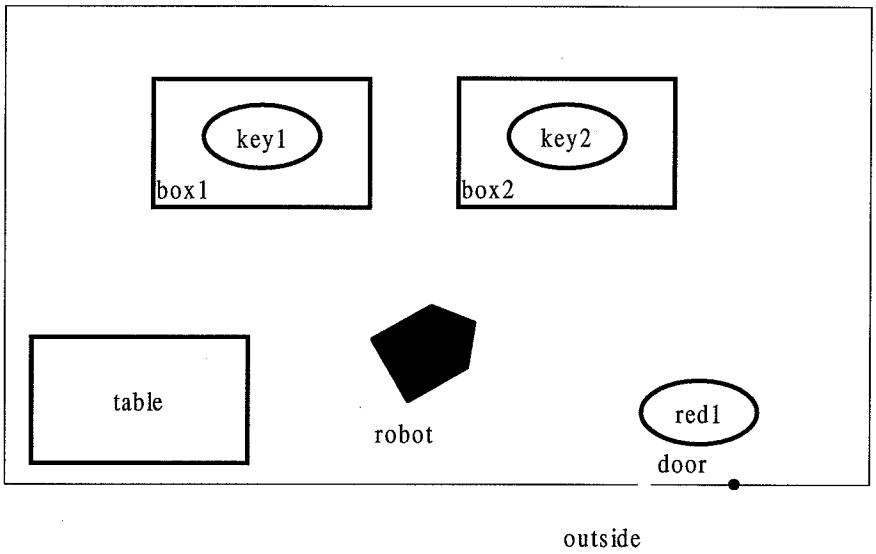
iv) Modify your program of b(iii) so that it neatly prints out the sequence of moves required to solve the game.

v) Describe (without necessarily giving the code) how efficiency could be improved by taking account of states which the system has discovered cannot lead to the goal.

The seven subparts carry, 10%, 10%, 10%, 40%, 10%, 10%, 10% of the marks.

2a Give the Prolog clauses (for `holds`, `poss_act`, `poss_state` and any others) that support finding a solution to a planning problem by simple depth-first search, based on the *situation calculus* representation of the problem.

b A *robot world* consists of two areas, "inside" and "outside", connected by a door. Inside there are four distinct locations: "table", "box1", "box2" and "door". There is also a robot inside, able to move about inside the room and transport objects from place to place. The robot can pick up an object from a location only if it is the sole object there. Furthermore, the robot cannot tell what, if anything, it has succeeded in picking up. At the start, box1 contains just "key1", box2 contains just "key2", there is just object "red1" by the door, and table is empty. In order for the robot to go or take anything outside, both key1 and key2 must be by the door. The goal is for the robot to take object red1 outside.



Represent this robot world in the situation calculus, giving Prolog clauses that express facts true at the start (about the location of the robot, where the objects are and the number of items at each location), facts that are always true, facts that need to hold in the goal state, facts established and facts invalidated by actions and the preconditions of actions.

HINT: You may wish to represent the *take* action by the term:

```
take(object, source, destination, n)
```

where n represents the number of items at the destination before the object being transported has been set down.

c If you were to employ the *WARPLAN* planning engine, rather than just depth-first search, what extra domain-specific information might you supply and how would it be used?

The three parts carry, respectively, 30%, 55%, 15% of the marks.

3a Give Prolog clauses for `between(i, j, k)`, which can be used both to test and generate an integer j lying between two given integers i and k, with $i \leq k$.

b The "long-form" of a date can be given as a Prolog list such as:

```
[28, th, november, 2015]
```

Give a set of Prolog grammar rules that define valid long-form dates between 1st January 2000 and 31st December 2099. Your rules should be able both to test and generate valid long-form dates.

c The "short-form" of a date can be given as a Prolog list such as:

[28, 11, 15]

The items on the short-form list may have a single digit. For example, [1, 1, 0] is the short-form of 1st January 2000.

By extending your grammar rules from part a, write a Prolog program that can translate between short-form and long-form valid dates between 1st January 2000 and 31st December 2099.

The three parts carry, respectively, 10%, 65%, 25% of the marks.

4a i) Give Prolog clauses for `sum_dist(t, s)` where t is a given list of numbers (some of which may occur more than once) and s is the sum of the distinct values on t.

ii) Define a finite domain constraint `diff_all(x, t)` which specifies that x is different from all the items on list t.

iii) Define a finite domain constraint `same_some(x, t)` which specifies that x is the same as at least one of the items on list t.

iv) Using ii) and iii), define a finite domain constraint `sum_distFD(t, s)` which specifies that s is the sum of the distinct values on the list t.

b i) Write a program in Prolog to solve the following puzzle:

The combination for a safe comprises three integers, each between 1 and 40. The second integer is twice the first. The third integer is 10 more than the second. The first integer has one even digit and one odd digit. The second integer has either two odd or two even digits. The total of all the distinct digits used in the combination is 18. What is the combination?

You do **NOT** need to solve the puzzle. In fact, the answer is: 14, 28, 38.

ii) Write a program in CLP(FD) to solve the same puzzle.

The six subparts carry, respectively, 10%, 10%, 10%, 10%, 30%, 30% of the marks.