

# TRINITY COLLEGE DUBLIN THE UNIVERSITY OF DUBLIN

**Faculty of Engineering, Mathematics and Science**

**School of Computer Science & Statistics**

**Integrated Computer Science Programme  
B.A. (Mod.) Business & Computing  
B.A. (Mod.) CSLL  
Mathematics  
Year 3 Annual Examinations**

**Trinity Term 2015**

## **Symbolic Programming**

**Thursday 21 May 2015**

**Goldsmith**

**9:30-11:30**

**Dr Tim Fernando**

---

### **Instructions to Candidates:**

Attempt **two** questions (out of the three given).

All questions carry equal marks. 50 marks per question.

You may not start this examination until you are instructed to do so by the Invigilator.

### **Materials permitted for this examination:**

Non-programmable calculators are permitted for this examination — please indicate the make and model of your calculator on each answer book used.

## 1. (a) Three translations of the English sentence

Mary owns every lamb that is white

into Prolog are

- (i) 'Mary owns every lamb that is white'.
- (ii) `own(mary,'every lamb that is white')`.
- (iii) `own(mary,X) :- lamb(X), white(X)`.

Which of (i)-(iii) are facts? Which are rules? Which translation is best and why?

[9 marks]

## (b) How does the Prolog interpreter respond to the following queries:

- (i) `3+2 = 5`.
- (ii) `3+2 = X`.
- (iii) `3+2 = 2+3`.
- (iv) `3+2 is X`.
- (v) `X is 3+2`.
- (vi) `[a|[b,c]] = [a,[b,c]]`.
- (vii) `[a,b|[c]] = [a|[b,c]]`.
- (viii) `[[a]] = [[a] | []]`.

[16 marks]

- (c) Define a 4-ary predicate `split` such that `split(N,List,Small,Big)` is true exactly when `List` is a list of numbers such that `Small` consists of all members of `List` less than `N` (occurring as many times in `Small` as in `List`), and `Big` consists of all members of `List` greater than or equal to `N` (occurring as many times in `Big` as in `List`).

For example,

```
| ?- split(3,[5,1,3,4],Small,Big).
Small = [1], Big = [5,3,4].
```

[10 marks]

- (d) Consider the binary predicate `sumOfPowers` such that `sumOfPowers(N, SoP)` is true exactly if  $N$  is a non-negative integer and  $SoP$  is the sum

$$\sum_{i=1}^N i^i = 1 + 2^2 + \dots + N^N$$

of powers  $i^i$  from  $i = 1$  to  $N$ .

For example, since  $1 + 2^2 + 3^3 = 32$ ,

```
| ?- sumOfPowers(3,S).
S = 32.
```

Define `sumOfPowers(N, SoP)` in Prolog.

For full credit, make sure your definition is tail-recursive.

[15 marks]

2. (a) Define the binary predicate `member` such that `member(X,List)` is true precisely if `X` is a member of `List`.

[5 marks]

- (b) Is the cut below red or green? Explain.

```
memb(X,[X|L]) :- !.
memb(X,[Y|L]) :- memb(X,L).
```

[10 marks]

- (c) We can define `first(X,List)` to be true precisely if `X` is the first element of `List` as follows

```
first(X,[X|_]).
```

Define `last(X,List)` to be true precisely if `X` is the last element of `List`.

[5 marks]

- (d) Define the binary predicate `multiple` such that `multiple(X,List)` is true precisely if `X` occurs at least twice in `List`. For example,

```
| ?- multiple(1,[2,1]).
no

| ?- multiple(1,[1,2,1]).
yes

| ?- multiple(1,[1,2,1,1]).
yes
```

[5 marks]

- (e) Define the 3-ary predicate `next` such that `next(A,B,List)` is true precisely if `List` is a list with `A` and `B` appearing consecutively in it.

For example,

```
| ?- next(A,B,[x,d,c]).
A = x, B = d ? ;
A = d, B = c ? ;
no.
```

[5 marks]

(f) Let the 6-ary predicate `mem3` be defined as follows

```
mem3(X1,X2,X3,L1,L2,L3) :- member(X1,L1),
                             member(X2,L2),
                             member(X3,L3).
```

Your task is to describe how Prolog backtracks on `mem3` by defining a 9-ary predicate

```
next3(A1,A2,A3,B1,B2,B3,L1,L2,L3)
```

that serves the role for `mem3(X1,X2,X3,L1,L2,L3)` that the predicate `next(A,B,List)` from part (e) serves for `member(X,List)`. More precisely, the requirement is that for all ground (i.e., variable-free) terms  $a1, a2, a3, b1, b2, b3, l1, l2, l3$ , we have `next3(a1,a2,a3,b1,b2,b3,l1,l2,l3)` true precisely if the query

```
| ?- mem3(X1,X2,X3,l1,l2,l3).
```

eventually returns the instantiations

```
X1 = a1, X2 = a2, X3 = a3
```

followed (on backtracking) immediately by

```
X1 = b1, X2 = b2, X3 = b3.
```

For example, since

```
| ?- mem3(X1,X2,X3,[1,2],[a],[x,y]).
X1 = 1, X2 = a, X3 = x ? ;
X1 = 1, X2 = a, X3 = y ? ;
X1 = 2, X2 = a, X3 = x ? ;
X1 = 2, X2 = a, X3 = y ? ;
no
```

we have `next3(A1,A2,A3,B1,B2,B3,[1,2],[a],[x,y])` true for the following three instantiations:

```
A1 = 1, A2 = a, A3 = x, B1 = 1, B2 = a, B3 = y
```

and

```
A1 = 1, A2 = a, A3 = y, B1 = 2, B2 = a, B3 = x
```

and

```
A1 = 2, A2 = a, A3 = x, B1 = 2, B2 = a, B3 = y.
```

[20 marks]

3. (a) Define a Definite Clause Grammar (DCG) for strings  $u2v$  where  $u$  and  $v$  are strings over the alphabet  $\{0, 1\}$  such that the number of 1's in  $u$  is *twice* the number of 0's in  $v$ . For example,

```
| ?- s([0,1,1,2,0,1,0],L).
L = [1,0];
L = [0];
no
```

[15 marks]

- (b) What are difference lists and how are they useful?

[5 marks]

- (c) Write your DCG in part (a) using ordinary Prolog notation, making the difference lists explicit.

[10 marks]

- (d) Write a DCG that given a non-negative integer *Half*, accepts a list of integers  $\geq 1$  that add up to twice *Half*. For example,

```
| ?- s(2,L,[]).
L = [4] ? ;
L = [3,1] ? ;
L = [2,2] ? ;
L = [1,3] ? ;
L = [2,1,1] ? ;
L = [1,2,1] ? ;
L = [1,1,2] ? ;
L = [1,1,1,1] ? ;
no
```

It may be useful to write a predicate `mkList(+Num,?List)` that returns a list *List* of integers from *Num* down to 1. For example,

```
| ?- mkList(3,L).
L = [3,2,1] ? ;
no
```

[20 marks]