Faculty of Engineering, Mathematics and Science
School of Computer Science & Statistics

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

Symbolic Programming

7 May 2016 Drawing Office 09:30 – 11:30

Dr Tim Fernando

Instructions to Candidates:
Attempt two questions. All questions carry equal marks. Each question is scored out of a total of 50 marks.

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) Specify Prolog’s response to the following queries.

   (i) X = 1.
   (ii) X == Y.
   (iii) 0+1 = 1+0.
   (iv) 0+1 =:= 1+0.
   (v) \( X \neq f(X) \).
   (vi) \( f(X) \neq g(Y) \).
   (vii) \([1|[2,3]] = .(1,.(2,[3])).\)
   (viii) \( X == f(X) \).
   (ix) \( X > 0 \).
   (x) \( \text{findall}(X, X \neq 1, L) \).

   [20 marks]

(b) Define a unary predicate isSet(+List) that is true exactly when no member of List occurs more than once in List. For example,

\[ \text{?- isSet([]).} \]
    yes
\[ \text{?- isSet([1,2]).} \]
    yes
\[ \text{?- isSet([1,1]).} \]
    no

   [10 marks]

(c) Define a unary predicate moreThanOne(+List) that is true exactly when List has more than one distinct member. For example,

\[ \text{?- moreThanOne([1,2]).} \]
    yes
\[ \text{?- moreThanOne([1,1]).} \]
    no

   [5 marks]
(d) Define a binary predicate moreThan(+List,+Num) that is true exactly when List is a list, Num is a non-negative integer and the number of distinct members of List is more than Num. For example,

\[
\begin{align*}
& \text{?- moreThan([1,2],1).} \\
& \text{yes} \\
& \text{?- moreThan([1,1,2],2).} \\
& \text{no}
\end{align*}
\]
2. (a) The *factorial* \( n! \) of a non-negative integer \( n \) can be defined as follows

\[
\begin{align*}
0! & := 1 \\
(n + 1)! & := n!(n + 1).
\end{align*}
\]

(i) The simplest translation of the recursive definition above into Prolog is *not* tail recursive. Use this to define a binary predicate \( \text{fac}(+N, ?\text{Factorial}) \).

[5 marks]

(ii) Write a tail recursive program for the factorial.

[10 marks]

(b) For a non-negative integer \( n \), the *nth Fibonacci number* \( F_n \) is defined as follows

\[
\begin{align*}
F_0 & := 0 \\
F_1 & := 1 \\
F_{n+2} & := F_n + F_{n+1}
\end{align*}
\]

giving \( F_2 = 0 + 1 = 1, F_3 = 1 + 1 = 2 \), etc.

(i) The simplest translation of the recursive definition above into Prolog is *not* tail recursive. Use this to define a binary predicate \( \text{fib}(+N, ?\text{Fibonacci}) \).

[10 marks]

(ii) Write a tail recursive program for the \( n \)th Fibonacci number.

[25 marks]
3. (a) Define a Definite Clause Grammar (DCG) for strings $a^n b^m c^k$ over the alphabet \{a, b, c\} where $0 \leq n, m, k$ and $n + m \leq k$. For example,

```
?- s([a,b,b,c,c,c,c],L).
L = [c] ;
L = [] ;
L = [a, b, b, c, c, c, c] ;
no.
```

[20 marks]

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

[5 marks]

(c) Write your DCG in part (a) with difference lists spelled out.

[10 marks]

(d) Write a DCG that given a list A and non-negative integer N, accepts a list of length $2* N$ of members of A. For example,

```
?- s([a,b],2,L,[]).
L = [a,a,a,a] ? ;
L = [a,a,a,b] ? ;
L = [a,a,b,a] ? ;
L = [a,a,b,b] ? ;
L = [a,b,a,a] ? ;
L = [a,b,a,b] ? ;
L = [a,b,b,a] ? ;
L = [a,b,b,b] ? ;
L = [b,a,a,a] ? ;
L = [b,a,a,b] ? ;
L = [b,a,b,a] ? ;
L = [b,a,b,b] ? ;
L = [b,b,a,a] ? ;
L = [b,b,a,b] ? ;
L = [b,b,b,a] ? ;
L = [b,b,b,b] ? ;
no.
```

[15 marks]