UNIVERSITY OF DUBLIN
TRINITY COLLEGE

Faculty of Engineering, Mathematics and Science

School of Computer Science & Statistics

Integrated Computer Science
B.A. (Mod.) CSLL
Mathematics

Trinity Term 2013

Symbolic Programming

Thursday 09/05/2013 **RDS, MAIN HALL(1500)** **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. Recall that we can encode the non-negative integers $0, 1, 2, \ldots$ as the terms `0`, `s(0)`, `s(s(0))`, ...

```
numeral(0).
numeral(s(X)) :- numeral(X).
```

with addition given by

```
add(0,X,X).
add(s(X),Y,s(Z)) :- add(X,Y,Z).
```

(a) Suppose we were to extend the knowledge base above with the clause

```
numeral(X+Y) :- numeral(X), numeral(Y).
```

Define a predicate `add2(X,Y,Z)` such that for instance,

```
?- add2(s(0)+s(s(0)), s(s(0)), Z).
Z = s(s(s(s(s(0))))) ;
no

?- add2(0, s(0)+s(s(0)), Z).
Z = s(s(s(0))) ;
no

?- add2(s(s(0)), s(0)+s(s(0)), Z).
Z = s(s(s(s(s(0))))) ;
no

?- add2(s(0)+s(0), s(0+s(s(0))), Z).
Z = s(s(s(s(s(0))))) ;
no
```

That is, the third argument `Z` of `add2(X,Y,Z)` can only be instantiated by `0`, `s(0)`, `s(s(0))`,..., and *not* `s(0+s(0))`, etc.

[15 marks]

(b) Next we introduce negative numbers via the function symbol `p` (for predecessor, -1, just as `s` stands for successor, $+1$).

```
numeral(p(X)) :- numeral(X).
```

Extend the predicate `add2` such that for instance,

```
?- add2(p(s(0)), s(s(0)), Z).
```

```
Z = s(s(0)) ;
no

?- add2(0, s(p(0)), Z).
Z = 0 ;
no

?- add2(p(0)+s(s(0)),s(s(0)),Z).
Z = s(s(s(0))) ;
no

?- add2(p(0), p(0)+s(p(0)), Z).
Z = p(p(0)) ;
no
```

Again, the third argument Z of add2(X,Y,Z) can be instantiated by only one ground (non-variable) term: either 0 or `s(0)` or `p(0)` or `s(s(0))` or `p(p(0))` or ... but *not* say, `s(p(0))`, etc.

[15 marks]

(c) Define a predicate `minus(X,Y)` such that for instance,

```
?- minus(0, Z).
Z = 0 ;
no

?- minus(s(s(0)), Z).
Z = p(p(0)) ;
no

?- minus(s(p(0)), Z).
Z = 0 ;
no

?- minus(p(s(p(0))), Z).
Z = s(0) ;
no
```

[10 marks]

(d) Let us extend `numeral` further to

```
numeral(-X) :- numeral(X).
```

Revise the predicate `add2(X,Y,Z)` such that for instance,

```
?- add2(-p(s(0)), s(s(0)), Z).
Z = s(s(0)) ;
no

?- add2(p(0)+s(s(0)), -s(s(0)), Z).
Z = p(0) ;
no
```

[5 marks]

(e) Define the predicate `subtract(X,Y,Z)` for subtracting `Y` from `X` to get `Z` such that for instance,

```
?- subtract(p(s(0)), s(s(0)), Z).
Z = p(p(0)) ;
no

?- subtract(p(0), -s(s(0)), Z).
Z = s(0) ;
no
```

[5 marks]

2. (a) Define a 3-ary Prolog predicate

```
if-then-else(A,B,C)
```

that returns B if A else C. Is your definition declarative or not? Explain.

[10 marks]

(b) Define a binary Prolog predicate

```
no-duplicates(List1,List2)
```

such that List2 is List1 with all duplications removed. For example,

```
?- no-duplicates([1,5,3,2,3,1,5,1],R).
R = [2,3,5,1]
```

[10 marks]

(c) Define a 3-ary Prolog predicate

```
minus(List1,List2,List3)
```

such that List3 is List1 with all members of List2 removed. For example,

```
?- minus([1,1,3,5,5,1],[3,3],R).
R = [1,1,5,5,1]
```

[10 marks]

(d) Given a binary predicate arc/2 between nodes, four different ways of defining when two nodes are connected by a non-empty sequence of arcs are given below.

```
connected1(A,B) :- arc(A,B).
connected1(A,B) :- connected1(X,B), arc(A,X).

connected2(A,B) :- arc(A,B).
connected2(A,B) :- arc(A,X), connected2(X,B).

connected3(A,B) :- arc(A,B).
connected3(A,B) :- connected3(A,X), arc(X,B).

connected4(A,B) :- connected4(A,X), arc(X,B).
connected4(A,B) :- arc(A,B).
```

(i) In what sense are the four predicates declaratively equivalent?

[5 marks]

(ii) Which of the four predicates for connectedness works best? Justify your answer.

[5 marks]

(iii) What problem does a fact such as `arc(a,a)` pose for all four predicates? How can one overcome this problem?

[10 marks]

3.

(a) What are difference lists and how are they useful.

[10 marks]

(b) Define a Defnite Clause Grammar (DCG) for the set of strings $a^n b^{n+m} c^m$ of length $2n + 2m$ for $n, m \geq 0$.

[20 marks]

(c) Write out the DCG you have in part (b) as an ordinary Prolog clauses, making the difference lists explicit.

[10 marks]

(d) Suppose we wanted to flatten a list to keep exactly the non-list members that appear in the list. For example,

```
?- flatten([1,[[],2,[3,4]],5,[6]], L).
L = [1, 2, 3, 4, 5, 6] ;
no
```

To avoid calling append (or `concatenate`), we can define `flatten` in terms of a 3-ary predicate `fl(List,L1,L2)` as follows

```
flatten(List,Flattened) :- fl(List, Flattened,[]).
```

Now, give a DCG for `fl(List,L1,L2)` that meets the specification above.

[10 marks]