Online Exam: CSU34011 Symbolic Programming
9 December 2021, 14:00-16:00 (+20 minutes for LENS students)

Instructions:
Answer two of the three questions, labelled Q1, Q2 and Q3. If you answer all three questions, your answer to question Q3 will be ignored (and left unmarked). Each question is worth 50 marks; two questions together are 100 marks.

Please type your answers and upload a plaintext file or PDF. (PDF converters are available online.)

If you have any questions or encounter any issues with the test, please contact `Tim.Fernando@tcd.ie`.

Tip: As partial credit will be awarded where possible, try *not* to leave any part of a question you are answering blank. Write down some thoughts that show effort and understanding (even if the understanding is that the effort falls short). But to avoid penalties for overly long answers, please be concise.

Declaration: By taking this exam, you are declaring

> *I understand that this is an individual assessment and that collaboration is not permitted.*
>
> *I have not received any assistance with my work for this assessment.*
>
> *Where I have used the published work of others, I have indicated this with appropriate citation.*
>
> *I have not and will not share any part of my work on this assessment, directly or indirectly, with any other student.*
>
> *I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at* `http://www.tcd.ie/calendar`.
>
> *I have also completed the Online Tutorial on avoiding plagiarism, located at* `https://libguides.tcd.ie/plagiarism/ready-steady-write`.
>
> *I understand that by taking this exam, I am agreeing with the statement above.*

[end of Declaration]

Question Q1

(a) Recall that the Prolog predicate =/2 is unification with**out** the occurs check. As a result, there are terms X and Y such that X=[0,X] and Y=[0|Y].

(i) Is the X such that X=[0,X] a list, and if so what are its members?

[5 marks]

(ii) Is the Y such that Y=[0|Y] a list, and if so what are its members?

[5 marks]

(b) Define a 3-ary predicate membe/3 so that membe(Item,List,Count) says Item occurs in List exactly Count times. For example,

```
?- membe(b,[a,b,a,a,b,c,c],N).
N = 2 ;
false
?- membe(d,[a,b,a,a,b,c,c],N).
N = 0 ;
false.
```

For full marks, keep your definition concise and make sure it also supports queries such as

```
?- membe(X,[a,b,a,a,b,c,c],2).
X = b ;
X = c ;
false.
```

[15 marks]

(c) Chapter 4 of *Learn Prolog Now* defines the binary relation sublist(SubL,L) to mean SubL is a prefix of some suffix of L.

```
sublist(SubL,L) :- append(_,S,L), append(SubL,_,S).
```

Consider the following variations of sublist.

```
sublis2(SubL,L) :- append(S,_,L), append(_,SubL,S).


sublis3(SubL,L) :- append(SubL,_,S), append(_,S,L).
```

(i) True or false: sublist and sublis2 give the same answers to every query.

[5 marks]

(ii) True or false: `sublist` and `sublis3` give the same answers to every query.

[5 marks]

(iii) Note that `sublist` can repeat a solution needlessly, as illustrated by the query

```
?- findall(S,sublist(S,[1,2]),L).
L = [[], [1], [1, 2], [], [2], []]
```

Revise `sublist` to a predicate `sub` that avoids repeated solutions so that for example,

```
?- findall(S,sub(S,[1,2]),L).
L = [[], [1], [1, 2], [2]]
```

For full marks, keep your code as concise as possible.

[10 marks]

(d) Suppose `length(L,N)` holds precisely when `L` is a list of length `N`. Suppose also that there are only finitely many solutions to the query

```
?- q(X).
```

True or false: the number of solutions to the query above is the instantiation of `N` that answers the query

```
?- findall(X,q(X),L), length(L,N).
```

Justify your answer.

[5 marks]

Question Q2

(a) Recall the built-in Prolog predicate `integer/1` such that for example,

```
?- integer(0).
true.
?- integer(1).
true,
?-integer(X).
false.
```

(i) Define a predicate `int/1` such that

```
?- int(X).
X = 0 ;
X = 1 ;
X = 2 ;
...
```

For full marks, ensure that all recursive predicates used to define `int` are tail-recursive.

[10 marks]

(ii) The predicate `integer/1` also holds for the negative integers. For example,

```
?- integer(-1).
true.
?- integer(-2).
true.
```

To enumerate every integer (0, positive and negative), define the Prolog predicate `intgr/1` so that

```
?- intgr(X).
X = 0 ;
X = 1 ;
X = -1 ;
X = 2 ;
X = -2 ;
X = 3 ;
X = -3 ;
...
```

[15 marks]

(b) Next, let us agree that a non-empty list of length *n* is *good* if it collects in reverse order the first *n* answers to the query

```
?- intgr(X).
```

For example, the lists `[0]` and `[1,0]` are good, as are the lists defined recursively by the Prolog predicate `good/1` as follows.

```
good([0]).
good([1,0]).
good([A,B|T]) :- good([B|T]), B>0, A is -B.
good([A,B|T]) :- good([B|T]), B<0, A is -B+1.
```

(i) If we try to enumerate every good list using the predicate `good` above, we run into trouble after the third solution:

```
?- good(L).
L = [0] ;
L = [1,0] ;
L = [-1,1,0] ;
Stack limit exceeded
```

Revise the definition of `good` above so that every good list is returned by the query `?- good(L).`

[10 marks]

(ii) Use DCGs to define a binary predicate `s/2` that generates every good list

```
?- s(L,[]).
L = [0] ;
L = [1,0] ;
L = [-1,1,0] ;
L = [2,-1,1,0] ;
L = [-2,2,-1,1,0] ;
...
```

[15 marks]

Question Q3

(a) Define `diff(A,B,C)` to hold exactly when `C` is the difference between lists `A` and `B` so that for example

```
? diff([a,b],[b],C).
C = [a] ;
false.

?- diff(A,B,C).
A = B, C = [] ;
A = [_1|B], C = [_1] ;
A = [_1,_2|B], C = [_1,_2] ;
...
```

[5 marks]

(b) Given a list `L = [`$a1, a2, \ldots, an$`]` of n elements, let us agree that an `L`-*sequence* is a list of length $n \cdot m$ for some integer $m \geq 0$ such that its first m elements are $a1$, its second m elements are $a2$, and ... its nth m elements are an. For example, if `L=[a,b]` then an `L`-sequence is a list of length $2m$ (for some integer $m \geq 0$) consisting of m `a`'s followed by m `b`'s. Your task is to capture `L`-sequences through a DCG for predicate `s/3` with an extra argument for `L`. For example,

```
?- s([a,b],S,[]).
S = [] ;
S = [a,b] ;
S = [a,a,b,b] ;
...

?- s([a,b,c],S,[]).
S = [] ;
S = [a,b,c] ;
S = [a,a,b,b,c,c] ;
...
```

[15 marks]

(c) For each integer $n > 0$, let L_n be the set of strings of the form $0^m 1^{m \cdot n}$ for some integer $m \geq 0$. For example, L_2 consists of the empty string, $011, 001111, \ldots$

(encoded in Prolog by the lists `[]`, `[0,1,1]`, `[0,0,1,1,1,1]`,...). Your task is to use DCGs to define a predicate `t/3` with an extra argument for n capturing L_n, as illustrated by

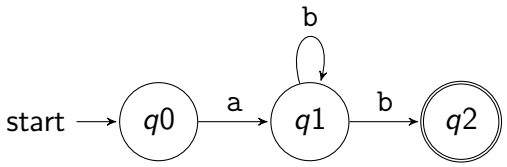
```
?- t(2,S,[]).
S = [] ;
S = [0,1,1] ;
S = [0,0,1,1,1,1] ;
...

?- t(4,S,[]).
S = [] ;
S = [0,1,1,1,1] ;
S = [0,0,1,1,1,1,1,1,1,1] ;
...
```

For full marks, keep your DCGs as concise as possible.

[15 marks]

(d) Recall from lecture that the finite automaton



(accepting the language ab^*b) can be encoded as the DCG

```
q0 --> [a], q1.
q1 --> [b], q1.
q1 --> [b], q2.
q2 --> [].
```

That is, with the DCG above, we get

```
?- q0([a,b],[]).
true.

?- q0([a,a,b],[]).
false.
```

Unfortunately, we loop on

```
?- q0(S,[]).
Stack limit exceeded
```

Fix the DCG above so that we get

```
?- q0(S,[]).
S = [a,b] ;
S = [a,b,b] ;
S = [a,b,b,b] ;
...
```

[15 marks]