ICS 1015/5002 Assignment 2018/19

Date due: 23rd January 2019

Your submission to the VLE should consist of five (5) parts, namely:

a) a single document which contains an explanation and the relevant code for all questions, each question in a separate section of the document (1 part);
b) a separate file (with .pl extension) with commented code for each question answered (4 parts).

In your document (see (a) above), for each question where coding is required, first write a concise description, in English, of your understanding of the problem and the logic you are going to use to solve it, and follow this by commented code that matches your description. Very important to note is that your ***explanations*** of the code you have written ***are as important as the code itself*** – so ensure that these are well written and clear!

Your submission must include, for each question, runnable code (see (b) above) that can be loaded (or copied and pasted) into the Prolog interpreter/compiler and run successfully – so do test it properly before submitting it, and make sure that it is very clear (by putting comments in the file) where the code for each problem begins and ends.

Please put your name in comments at the top of each file.

/* … and use comments to explain what variable names stand for or what particular lines of code do! */

If you are unable to upload your submission to the VLE, then please contact me or the departmental office to see how best to submit your assignment.

If you have any questions, please do not hesitate to contact me by email using this address: peter.xuereb@um.edu.mt

1) Write a prolog program that, using data requested from the user, calculates the areas of a circle, square, rectangle, triangle, and parallelogram, and the volumes of a sphere, cube, cyclinder and cone. Wherever possible, reuse code already written. Explain the dialog with the user, and show a number of test cases.

[20 marks]

2) a) Write a predicate extract_ints/2 to extract integers from a list.

(Hint: you can use the system predicate integer/1 for this – see the help file).

Your output should look like this:

```
| ?- extract_ints([a,b,c,3,2,5,z,[4,3,2],x,6,4,b,d],L).
L = [3,2,5,6,4]
```

[7 marks]

b) Write a predicate remove_dups/2 to remove duplicates from a list.

(Hint: you can use the system predicate member/2 for this – see the help file).

Your output should look like this:

```
| ?- remove_dups([a,b,c,a,b,[1,2],b,a,[1,2],z],L).
L = [c,b,a,[1,2],z]
```

[7 marks]

c) Using system predicate append/3, write predicates to return:

i. the last element of a list
ii. the last 2 elements of a list
iii. the last n elements of a list
iv. the first n elements of a list.

(Hint: you can use the system predicate length/2 for this – see the help file).

[6 marks]

[7+7+6 = 20 marks]

3. Given that a sublist is defined as *'a list that makes up part of a larger list'*, give a few cases where, according to this definition, the system predicate sublist/2 does not work correctly.

   The following code fixes the issue:

```
subl(L1,L2):-subl(L1,L1,L2).

subl(OS,[X],[X|T]):-write('found '),write(OS),write(' at front of '),append(OS,T,T1),write(T1),nl.
subl(OS,T,[]):-fail.

subl(OS,[H1,H2|T1],[H1,H2|T2]):-subl(OS,[H2|T1],[H2|T2]).
subl(OS,[H1,H2|T1],[H1,H3|T2]):-subl(OS,OS,[H3|T2]).
subl(OS,[H1|T1],[H3|T2]):-subl(OS,OS,T2).
```

   a) Why does subl/2 call subl/3? **[2]**
   b) What is OS and why is it used? **[3]**
   c) Take a specific call to subl/2 -- `subl([2,3,[3,4]],[1,2,3,4,2,3,[3,4],4,5])` -- and run through it in the debugger, explaining how it works, and taking care to explain the use of OS during the call sequence. **[15]**

```
| ?- subl([2,3,[3,4]],[1,2,3,4,2,3,[3,4],4,5]).
found [2,3,[3,4]] at front of [2,3,[3,4],4,5]
yes
```

[2+3+15 = 20 marks]

4. Below are two sorting algorithms.

```
asort(Xs, Ys):-asort1(Xs, [], Ys),!.

asort1([], Ys, Ys).
asort1([X|Xs],Ys0,Ys):-insert(Ys0,X,Ys1),asort1(Xs,Ys1,Ys).

insert([Y|Ys], X, [Y|Zs]):-Y < X, !, insert(Ys, X, Zs).
insert(Ys, X, [X|Ys]).
```

```
bsort([X|Xs],[Y|Ys]):-minimum(Xs,X,Y),efface([X|Xs],Y,Zs),bsort(Zs,Ys),!.
bsort([], []).

minimum([Y|Ys],X,Z):- Y =< X, !,minimum(Ys,Y,Z).
minimum([_|Ys],X,Z):- minimum(Ys,X,Z).
minimum([],X,X).

efface([Y|Xs],Y,Xs):-!.
efface([X|Xs],Y,[X|Zs]):-efface(Xs,Y,Zs).
```

   The output from these algorithms looks like this:

```
| ?- asort([4,2,3,5,48,24,67,6,4,2],L).
L = [2,2,3,4,4,5,6,24,48,67]

| ?- bsort([4,2,3,5,48,24,67,6,4,2],L).
L = [2,2,3,4,4,5,6,24,48,67]
```

a) Add write statements to appropriate places in the code of both algorithms to illustrate their operation. **[4]**
b) Compare the way the two sorting algorithms work. **[12]**
c) From the way they manipulate the elements in their lists, identify the type of sorting algorithms that they are. **[4]**

[4+12+4 = 20 marks]

d) Pick any sorting algorithm (except from those used previously in this assignment) implemented in prolog (from the internet or otherwise) and explain, using the trace function and appropriate examples, how the algorithm works.

[20 marks]