CPSC 312

Assignment #2 - List Processing

Due no later than 11:59pm on Tuesday, October 6th, 2015.

Please include your official full name, your student id as well as your ugrad id in your submission.

How to hand-in: This assignment will be submitted online, but I have not set up the server yet, so do not submit anything until you hear the announcement on how to do so.

Collaboration and Academic Honesty: This assignment is meant to be done individually. This means no consulting with other students as well as other sources. The various exercises given in the class are your chance to discuss similar questions with your peers but the assignment must be a solo effort. The solution to most of these problems are not difficult to find. However, the purpose of this assignment is to prepare you for the midterm and the final exams where you will have to write similar programs without the aid of the internet, various logic programming textbooks, and your peers. So make sure you are using this opportunity to prepare yourself.

Each of the following questions asks you to write a prolog procedure (rule or predicate) that process lists. If you are making references to other procedures in your code, whether it's an additional procedure you've written or it's a procedure we've seen in the lecture, or a built-in SWI-Prolog procedure, you need to include the code for those as well in your solution. In other words, make sure what you submit works as a complete and independent prolog program.

Additionally, for every procedure, you need to include
1) The Induction steps (as seen in the lecture) to show how you got to the solution. If you didn't use induction or don't want to include the steps, write in simple English how your recursive procedure works to produce the expected result: simply stating what the base case represents and what recursive step is taken.
2) the output of your procedure when run for the cases given as part of the question.

A code that is not accompanied by the above mentioned documentation will not receive any mark. It is best to include the documentation as a separate txt or doc file.

Try your best to write the smallest, and the least redundant procedures that you can. If your code works but has significant redundancies, you'll receive a lower mark.

All questions are expected to be solved through recursion and with no use of numbers or arithmetic procedures. It's okay however, to write auxiliary procedures that carry out part of the recursion (if necessary), rather than one main recursive procedure.

-----------------------------------------------------------------

(10) Problem 1: Write a procedure

```
proc1(L1, L2, L3)
```

which returns true if list L3 is the result of combining lists L1 and L2 such that the first element of L3 is the first element of L1, the second element of L3 is the first element of L2, the third element of L3 is the second element of L1, and so on. For example,

```
proc1([a,b,c],[d,e,f],[a,d,b,e,c,f])
```

returns true. If L1 and L2 do not have an equal number of elements, your procedure should trail the extra elements from the longer list at the end of the resulting list. Here is an example:

```
 proc1([a,b],[d,e,f,g],[a,d,b,e,f,g])
```

This should also work:

```
  ?- proc1(X,Y,[1,2,3,4,5,6]).
X = [1, 2, 3, 4, 5, 6],
Y = [] ;
X = [],
Y = [1, 2, 3, 4, 5, 6] ;
X = [1, 3, 4, 5, 6],
Y = [2] ;
X = [1],
Y = [2, 3, 4, 5, 6] ;
X = [1, 3, 5, 6],
Y = [2, 4] ;
X = [1, 3],
Y = [2, 4, 5, 6] ;
X = [1, 3, 5],
Y = [2, 4, 6] ;
X = [1, 3, 5],
Y = [2, 4, 6] ;
false.
```

----------------------------------------------------------------

Problem 2: Write a procedure

```
  proc2(L1, L2)
```

which

a. (10) returns true if L2 is a duplication of L1 according to the pattern in the following example:

```
?- proc2a([a,b,c],[a,a,b,b,c,c]).
true.
```

Note that the elements that are duplicated are also adjacent to each other and appear in the same order in both lists.

These cases should also work:

```
?- proc2a(X,[a,a,b,b,c,c]).
X = [a, b, c].

?- proc2([a,b,c],X).
X = [a, a, b, b, c, c].
```

This case should not work:

```
?- proc2a([a,b,c],[a,b,c,a,b,c]).
false
```

b. (10) returns true if every element in list L1 appears once or more in L2 in any order, adjacent or not (and L2 has no elements that’s not in L1); such as

```
?- proc2b([a,b,c],[a,b,a,b,c,c,c]).
true.
```

This should also work:

```
?- proc2b(X,[a,b,a,b,c,c,c]).
X = [a, b, c].
```

------------------------------------------------------------------

(10) Problem 3: Write a procedure

```
proc3(L1, L2)
```

which returns true if list L2 is the result of removing all duplicate elements from list L1. For example,

```
proc3([a,b,c,b,d,b],[a,c,d,b])
```

returns true (note that the last duplicate 'b' is the one that remains). This should also work:

```
?- proc3([a,b,c,b,d,b],X).
X = [a, c, d, b] ;
X = [a, c, d, b] ;
false.
```

-----------------------------------------------------------------

(10) Problem 4:  Write a procedure

```
proc4(L1, L2)
```

which returns true if lists L1 and L2 contain exactly the same elements, although possibly in different order.  For example,

```
proc4([a,b,c],[b,c,a])
proc4([a,b,c],[a,c,b])
proc4([a,b,c],[c,b,a])
```

all return true.  This should also work:

```
?- proc4([a,b,c],X).
X = [a, b, c] ;
X = [b, a, c] ;
X = [b, c, a] ;
X = [a, c, b] ;
X = [c, a, b] ;
X = [c, b, a] ;
false.
```