Lab Exercises - Cut and Fail

CMPT333N

Problem 1

Say what the effects would be for each of the goals

```
?- f(p).
?- f(q).
?- f(r).
?- f(X).
```

if the program loaded in was the following:

1. ```
   f(X) :- !, X = p.
   f(X) :- !, X = q.
   f(X) :- X = r.
   ```

2. ```
   f(X) :- X = p, !.
   f(X) :- X = q, !.
   f(X) :- X = r.
   ```

3. ```
   f(X) :- X = p, !.
   f(X) :- !, X = q.
   f(X) :- X = r.
   ```

4. ```
   f(X) :- !, X = p.
   f(X) :- X = q, !.
   f(X) :- X = r.
   ```

5. ```
   f(X) :- X = p.
   f(X) :- X = q, !.
   f(X) :- X = r.
   ```

6. ```
   f(p) :- !.
   f(q) :- !.
   f(r).
   ```

7. ```
   f(p).
   f(q):- !.
   f(r).
   ```
   ```

Problem 2

Given the following program:

```
unmarried_student(X):-
    not(married(X)), student(X).

student(joe).
married(john).
```

This program seems to suggest that joe is an unmarried student, and that joe is not an unmarried student, and indeed:

```
?- unmarried_student(joe).

yes
?- unmarried_student(john).

no
```

But, for logical consistence, asking for unmarried students should return joe as answer, and this is not what happens:

```
?- unmarried_student(X).

no
```

The reason for this is that the call to `not(married(X))` is not returning the students which are not married: it is just failing because there is at least a married student.

Change the `unmarried_student/1` predicate so that it works correctly in the three queries shown above.

Problem 3

Analyze the `my_member/2` program below:

```
my_member(Item, [Item|_]).

my_member(Item, [_|Tail]):-
    my_member(Item, Tail).
```

This program can be used to discover whether the first argument was an element of the second argument, but, if the first argument is a variable, it can also be used to generate a set of answers:

```
?- my_member(X, [a,b,c]).
X = a ? ;
X = b ? ;
X = c ? ;
no
```

Here is a modified version of the of `my_member/2`; one with a cut in the first clause. How does its behaviour differ from the previous definition, and why?

```
my_memberchk(Item, [Item|_]):- !.

my_memberchk(Item, [_|Tail]):-
    my_memberchk(Item, Tail).
```

Problem 4

Consider the following program:

```
p(Y) :- q(X,Y), r(Y).
p(X) :- q(X,X).
q(a,a).
q(a,b).
r(b).
```

Introduce the cut operator (!) in different places of this code, and comment on the results for query `p(Z)`.

Problem 5

What does the program below implement?

```
c(_, [], 0).
c(X, [X|L], N):-
    c(X, L, R), !,
    N is R+1.
c(X, [_|L], N):-
    c(X, L, N).
```