<u>Question 1</u> [2 marks]

Given the following program:

```
postIt([]).
postIt([c|R]):- postIt(R), !, nl.
postIt([X|R]):- postIt(R), write(X).
```

What will be printed as response to the following query?

```
?- postIt([a,b,c,d,e]).
```

<u>Question 2</u> [3 marks]

Complete the predicate negCount below such that it counts the negative numbers in a list, e.g.,

```
?- negCount([0,4,-3,-1,6,-7], N).
N = 3
```

Note: You are not allowed to change the order of the following rules.

```
negCount([],0).



negCount([X|L],N) :- ____________________________________________


negCount([X|L],N) :- X>=0, negCount(L,N).
```

Question 3 [2 marks]

The following predicate q3 below is designed to operate on binary trees:

```
q3(t(V, nul, nul), 0).
q3(t(V, Q, nul), 1).
q3(t(V, nul, Q), 1).
q3(t(V, Q1, Q2), T) :- q3(Q1, T1), q3(Q2, T2), T is 1+T1+T2.
```

What value for T is obtained with the following query?

```
?- q3(t(4,
              t(2,
                   nul,
                   t(3, t(1,nul,nul), t(9,nul,nul))),
              t(7, t(5, nul, t(6, nul, nul)),
                   t(9, t(1,nul,nul), t(9,nul,nul)))),T).
```

T=

<u>Question 4</u> [4 marks]

The following facts describe which license or permit is held by whom. The list includes driving licenses, fishing permits and licensed weapons.

```
permitted(robert,fishing).
permitted(jochen,driving).
permitted(paul,fishing).
permitted(jean,weapons).
permitted(jean,driving).
permitted(sam,weapons).
permitted(sam,fishing).
```

a) Give a query which finds a person who is **not** permitted to drive.

b) List in order **all solutions** found by the following query.

```
?- permitted(X,Y),permitted(X,Z),Y\==Z.
```

Question 5 [5 marks]

a) Given the following Prolog program

```
p(X) :- b(X), c(Y).
p(X) :- a(X).
c(X) :- d(X).
a(1).
a(2).
a(3).
b(4).
b(5).
d(6).
d(7).
```

Draw the complete Prolog search tree for the following query (clearly mark the solutions found and the **order** in which they are found).

```
?- p(X).
```

Question 5 (continued)

b) List the solutions which are found by the same query when a Cut is added as below:

```
p(X) :- b(X), !, c(Y).
p(X) :- a(X).
c(X) :- d(X).
a(1).
a(2).
a(3).
b(4).
b(5).
d(6).
d(7).
```

c) List the solutions which are found by the same query when a Cut is added as below:

```
p(X) :- b(X), c(Y).
p(X) :- a(X).
c(X) :- d(X).
a(1).
a(2):- !.
a(3).
b(4).
b(5).
d(6).
d(7).
```

<u>Question 6</u> [2 marks]

Which of the predicates below works correctly? The predicate is to substitute all elements of the list equal the first argument with the second argument. For example:

```
?- subElement(apple, orange, [apple, celery, pear, pear, apple, raisin],L).
     L = [orange, celery, pear, pear, orange, raisin]
```

a) subElement(_,_,[],[]). subElement(X,Y,[X|R],[Y|R]) :- subElement(X,Y,R,R). subElement(X,Y,[Z|R],[Z|R]) :- X\==Z, subElement(X,Y,R,R).	**b)** subElement(_,_,[],[]). subElement(X,Y,[X|R],[Y|R1]) :- subElement(X,Y,R,R1). subElement(X,Y,[Z|R],[Z|R1]) :- X==Z, subElement(X,Y,R,R1).
c) subElement(_,_,[],[]). subElement(X,Y,[Z|R],[Z|R1]) :- subElement(X,Y,R,R1). subElement(X,Y,[X|R],[Y|R1]) :- X==Z, subElement(X,Y,R,R1).	**d)** subElement(_,_,[],[]). subElement(X,Y,[X|R],[Y|R1]) :- subElement(X,Y,R,R1). subElement(X,Y,[Z|R],[Z|R1]) :- X\==Z, subElement(X,Y,R,R1).
e) subElement(_,_,[],[]). subElement(X,Y,[Z|R],[Z|R1]) :- subElement(X,Y,R,R1). subElement(X,Y,[X|R],[Y|R1]) :- X\==Z, subElement(X,Y,R,R1).	**f)** subElement(_,_,[],[]). subElement(X,Y,[X|R],[X|R1]) :- subElement(X,Y,R,R1). subElement(X,Y,[Z|R],[Z|R1]) :- X\==Z, subElement(X,Y,R,R1).

Question 7 [6 marks]
Given the following database:

```
prerequisite(csi2520,csi2510).
prerequisite(csi2520,csi2610).
prerequisite(csi2510,iti1521).
prerequisite(csi2510,mat1748).
prerequisite(csi2510,csi2772).
```

What is the value of L obtained by each of the following queries (if multiple solutions are possible, list only the first solution that will be found)?

```
?- bagof(X,Y^prerequisite(X,Y),L).
```

L=

```
?- setof(X,Y^prerequisite(X,Y),L).
```

L=

```
?- setof(Y,prerequisite(X,Y),L)
```

L=