Question 1 [6 marks]

Given the following program:

```
rail('Jasper','Prince George').
rail('Prince George', 'Quesnel').
rail('Quesnel','Whistler').
rail('Whistler','Vancouver').
rail('Vancouver','Kamloops').
rail('Kamloops','Jasper').
rail('Kamloops','Banff').
rail('Banff','Calgary').
rail('Calgary','Edmonton').
rail('Edmonton','Jasper').

ride(X,Y) :- rail(X,Y).
ride(X,Y) :- rail(Y,X).

train_ride(X,Y) :- write(X),write(' '),
              train(X,Y,[X]), !.


train(X,Y,_) :- ride(X,Y),
           write(Y),!.

train(X,Y,L) :- ride(X,Z),
           \+member(Z,L),
           write(Z),write(' '),
           train(Z,Y,[Z|L]).
```

What will be printed as response to the following query?

```
?- train_ride('Vancouver','Quesnel').
```

Question 2 **[3 marks]**

Jane likes apps that have 3D graphics and/or are coded in Prolog but she does not like slow apps. Given the rule base:

```
app(mail).
app(calendar).
app(image_viewer).
app(browser).
graphics3D(image_viewer).
slow(calendar).
coded(mail,prolog).
coded(browser,go).
coded(calendar,prolog).

likes(jane,X) :- app(X), neat(X).

neat(X) :- graphics3D(X).
neat(X) :- coded(X,prolog).
```

Consider the following query:

```
1 ?- likes(jane,X).
X = mail ;
X = calendar ;
X = image_viewer ;
false.
```

It produces the incorrect result X = calendar, fix the neat predicate to work as desired with a cut (!).

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

```
likes(jane,X) :- app(X), neat(X).


neat(X) :- ______________________________________________


neat(X) :- graphics3D(X).
neat(X) :- coded(X,prolog).
```

<u>Question 3</u> [4 marks]

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

```
tree(X) :- X =
        t(25,
          t(20,
            t(12,
              t(5,nil, nil),
              t(15,nil,nil)),
            t(23,nil,nil)),
          t(28,
            t(24, nil, nil),
            nil))

traversal(nil).

traversal(t(Root,Left,Right)) :-
    traversal(Left),
    traversal(Right),
    write(Root),
    write(' ').

rmm(t(M, Left, nil), Left, M).
rmm(t(Root, Left, Right),
    t(Root, Left, RightS), M) :- rmm(Right, RightS, M).
```

a) What is printed with the following query with the following query?

```
?- tree(X), traversal(X).
```

b) What is the value of Z printed by the following query?

```
?- tree(X), rmm(X,Y,Z).
```

<u>Question 4</u> [5 marks]

a) Given the following Prolog program

```
artist( 'Lucas' ).
artist( 'Charlotte' ).
developer( 'Mia' ).
developer( 'Sophia' ).
developer( 'Mason' ).
game( 'Minecraft' ).
game( 'Doom' ).
game( 'Pong' ).

team( A, B ) :- artist(A),
                developer(B).
team( A, B ) :- developer(A),
                artist(B).

design( A, B, C ) :- team( A, B ),
                not(game( C )).
```

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

```
?- design( 'Charlotte', D, 'Secret').
```

(please answer on the next page)

b) Insert a single cut (!) in the program above such that whenever Sophia is selected as a developer, no other developer will be considered during backtracking.

(Provide your answer for question 4a) here.)

<u>Question 5</u> [2 marks]

Which of the predicate(s) below work(s) correctly? The predicate is to replace every entry in the list with the sum up to and including the current element. For example:

```
?- rSum( [2,3,7,2], R ).
R = [2, 5, 12, 14].
```

a)

```
rSum( L, R  ) :- rSum( L, 0, [], R ).


rSum( [], _ , R, R) :- !.
rSum( [H|T], S, L, R ) :-
       RH is S+H,
       rSum( T, RH, [RH|L], R ).
```

b)

```
rSum( L, R  ) :- rSum( L, X, R ).

rSum( _, 0 , []).
rSum( [H|T], S, [RH|RT] ) :-
       RH is S+H,
       rSum( T, RH, RT ).
```

c)

```
rSum( L, R  ) :- rSum( L, X, R ).


rSum( [], _ , []).
rSum( [H|T], S, [RH|RT] ) :-
       RH is S+H,
       rSum( T, RH, RT ).
```

d)

```
rSum( L, R  ) :- rSum( L, 0, R ).


rSum( [], _ , []).
rSum( [H|T], S, [RH|RT] ) :-
       RH is S+H,
       rSum( T, RH, RT ).
```

e)

```
rSum( L, R  ) :- rSum( L, 0, R, R ).


rSum( [], _ , R, R) :- !.
rSum( [H|T], S, L, R ) :-
       RH is S+H,
       rSum( T, RH, [RH|L], R ).
```

f)

```
rSum( L, R  ) :- rSum( L, 0, [], R ).


rSum( _, [] , [], []) :- !.
rSum( [H|T], S, L, R ) :-
       RH is S+H,
       rSum( T, RH, [RH|L], R ).
```

Question 6 [6 marks]

Given the following rule base:

```
vegetarian( 'Salad rolls', 8.5 ).
meat( 'Hamburger and fries', 10.0 ).
meat( 'Cheeseburger and ceasar salad', 12.5 ).
meat( 'Steak on a bun', 12.5 ).
fish( 'Fish and chips', 12.5 ).

meal( X, P ) :- vegetarian( X, P ).
meal( X, P ) :- meat( X, P ).
meal( X, P ) :- fish( X, P ).

orderForTwo( X,Y,P ) :- meal( X, P1 ),
                meal( Y, P2 ),
                P >= P1 + P2.
```

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)?

```
?- findall( X, orderForTwo( X,Y,18.5), L ).
```

L=

```
?- setof( X, Y^orderForTwo( X,Y,18.5), L ).
```

L=

```
?- bagof( (X,Y), orderForTwo( X,Y,18.5), L ).
```

L=