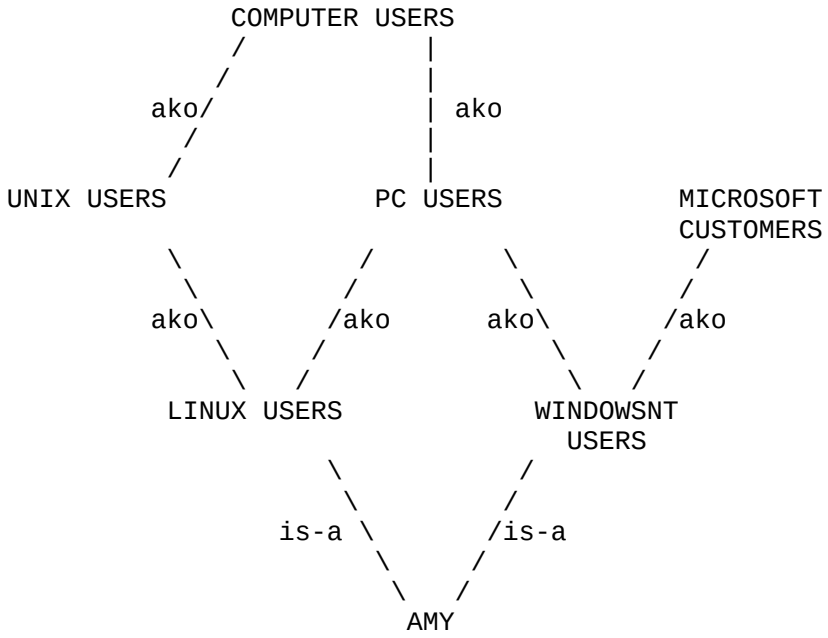
Question1

- [2Marks]



1. Give the class-precedence list for Amy that would be obtained by applying the topological-sorting algorithm to the previous graph (1Marks)

Class Precedence list :
Amy
Linux Users
Unix Users
WindowsNT Users
PC Users
Computer Users
Microsoft Customers

2. Suppose that each of the classes *Unix users*, *PC users* and *Computer Users* contains a *favorite programming language* slot. The default value for this slot is:
    3. Java, for the *Computer Users* class.
    4. C#, for the *Unix Users* class.
    5. C++, for the *PC Users* class.

   What is the value obtained for Amy's favorite programming language according to the class-precedence list you constructed above? Explain you answer. (1Marks)
   According to the class-precedence list

   Amy's favorite programming language is C#

- [2Marks]

KR scheme	Inference1	Inference2	Inference3
Frames	Multiple-Inheritance		
Semantic Networks	Intersection search	Inheritance	
First Order Logic	Resolution	Resolution by refutation	unification

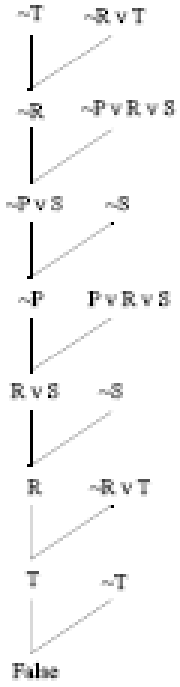
Question2

Consider the following 4 premise sentences in PL:

1. $P \Rightarrow (R \vee S)$,
2. $\neg P \Rightarrow (R \vee S)$,
3. $\neg S$,
4. $(R \vee U) \Rightarrow T$

(a) Convert the premise sentences into conjunctive normal form (CNF) and show the result as a set of **clauses**. [1Mark]

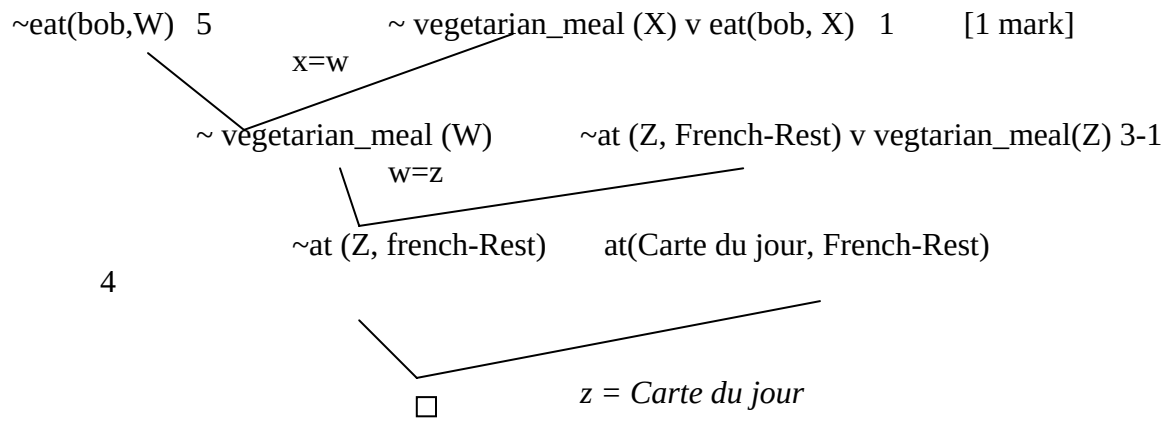
1. $\neg P \vee R \vee S$,
2. $P \vee R \vee S$,
3. $\neg S$,
4. $\neg R \vee T$,
5. $\neg U \vee T$

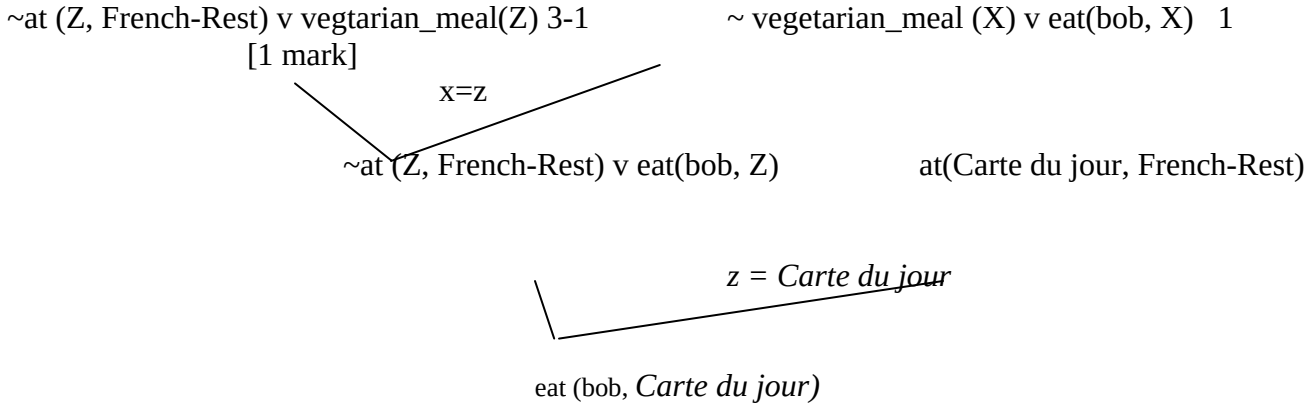


1. (∀X) vegetarian_meal (X)-> eat(bob,X)
2. (∀Y) vegetarian_meal(Y)-> ~contain(Y, meat)
3. (∀z) meal(Z) ^ at(Z, french-rest) -> vegtarian_meal(Z)
4. at(carte du jour, french-rest)

5- Goal eat(bob,W)

1- ~ vegetarian_meal (x) v eat(Bob, x) [1Mark]
2- ~vegetarian_meal(y) v ~contain(y,meat)
3-1 ~at (z, French-Rest) v vegtarian_meal(z)
3-2 ~meal(Z) v vegtarian_meal(Z)
4- at(Carte du jour, French-Rest)





```
next_oldest(ann,bern).
next_oldest(bern,carmen).
next_oldest(carmen,diane).
next_oldest(diane,emil).
next_oldest(emil,fiona).
next_oldest(fiona,george).

older(X,Y) :- next_oldest(X,Y).
older(X,Z) :- next_oldest(Y,Z), older(X,Y).
```

(a) What would be Prolog's answer to the query next oldest(carmen,fiona).
No

(b) What would be Prolog's answer to the query older(ann,bert).
yes

(c) List Prolog's answer to the query older(X,diane) in the order that they are produced.
X = carmen, X = bern, X = ann, no

(d) List Prolog's answer to the query `older(emil,Y)` in the order that they are produced.
Y = fiona, Y = george, no