CSCI 320, Homework 10, Due by **Tuesday, December 6, noon** (upload to OAKS).
Logic programming in Prolog

Collaboration: You may discuss the assignment with peers, but you must do your own work. It is not OK to copy or share solutions, or look for solutions on the Internet. If you do run into a solution, you may use it but you must acknowledge the find and you may not share it with others. It **is OK** to discuss how to use the prolog editor and prolog system in the lab, which is now working . Log in using your Woosters account and put input into the editor window – issue a "consult" command when you want your code to be executed. The editor comes with a manual and automatically finds the prolog interpreter.

1-4: Exercises 1-4, Sebesta, page 736. Use the following names for the main rules in each:
a- `maximum`, 2 – `intersection`, 3- `union`, 4 –`last`

5- Define the `nonMember` relationship, `nonMember`(X, List) that does not use the member relationship. nonMember returns true if element X is not in list List.

6- Write a program call `fib`, such that `fib`(n, fn) is true if and only if fn is the nth number in the Fibonacci sequence. Recall that $fib_0 = 1$, $fib_1 = 1$ and for all n>1, $fib_n = fib_{n-1} + fib_{n-2}$.

7- Given a list of the form [`item`(name1, price1), `item`(name2, price2),..., `item`(name*k*, price*k*)].

a. Write rule(s) called `cost`(L, Name, Price) which given a list L (as above) and Name, gives the Price of that item.
b. Can this rule be run backward? That is, what happens when you provide the list L and Price, but not the Name?
c. Write rule(s) called `listcost`(L, Names, Price) where L is a list as above, Names is a list of names of items, and Price is to be computed as the total cost of all the items in list Names. Can this be run backward?

8 – Consider a graph represented as a sequence of edges between vertices such as below (written more than one per line to save space here):

```
edge(a, b).    edge(a, f).    edge(b, c).    edge(b, f).
edge(c, a).    edge(c, f).    edge(d, e).    edge(e, a).
edge(e, c).
```

Write rule(s) that give a path from a start vertex to a finish vertex, `path(Start, Finish).` The path that may not visit any vertex more than once, other than possibly the start vertex, which could also be the finish vertex. For the graph above, one path from a to f would be a – b – f, another is a-f, but the following a – b – c –a – f is not considered a path for this problem.

All code and non-code answers must appear in a single file, which is to be uploaded to OAKS. For non-code answers, include them as comments ('%' at start of line). Make sure that documentation includes your name and a label in terms of problem number and letter, if appropriate, for each problem. Make sure that they are in order in the file uploaded.