CMSC 671 **Artificial Intelligence** Fall 2014
Section 01 TuTh 1:00 - 2:15pm IT227

Project 1

Due Sept. 23, 2014

This project is an exercise of Lisp programming. You are asked to write several Lisp functions for some simple tasks, and apply your code to given test data.

1. (20 points) Fibonacci numbers are defined recursively as follows:

$$Fib(n) = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ Fib(n-1) + Fib(n-2) & \text{if } n > 1. \end{cases}$$

   Write an *efficient* Lisp function ***fib*(*n*)** that generates the sequence of Fibonacci numbers Fib(0),…Fib(n).

   Test your code with **n = 1, n = 5,** and **n = 10**.

2. (20 points) Consider a list L each of whose elements consists of its name (a string) and value (an integer), represented as a (name value) list. Elements in L are in ascending order of their values. Write a Lisp function to modify L by inserting a new element x into list L in such a way that the resultant list remains in the ascending order of their elements' values. For simplicity, you can assume that no two elements will have the same name, and elements with different names always have different values.

   Test data:
   Initial list: L = ((“a” 2) (“d” 3) (“c” 10));
   Elements to be inserted: 1) (“b” 5) 2) (“e” 20) 3) (“f” 1) 4) (“g” 6)

   Output: The final L list after successive insertions of the four new elements given above.

3. (20 points) Write a Lisp function, called ***modify***, with three arguments S1, S2 and S3. This function replaces every occurrence of S1 in S2 by S3. S2 remains unchanged if S1 does not appear in S2. For example, (modify ‘a ‘(a (b a)) ‘(1 2)) should return a list ((1 2) (b (1 2)). Note that 1) the structures of the three S-expressions S1, S2 and S3 can be arbitrarily complex, and 2) we do not consider (a b) as occurring in (a b c) while it is occurring in ((a b) c).

   Test data:

1)	S1 = a,	S2 = nil,	S3 = (x)
2)	S1 = c,	S2 = (((a (b c)) c))	S3 = (a b)
3)	S1 = (a b),	S2 = (a (b (a b) c) (a b))	S3 = (x y)
4)	S1 = (u),	S2 = ((a (u v)) ((u)) (y (u) v))),	S3 = v.

4. (40 points) You are asked to write two functions related to the 8-queen problem as defined in the textbook. Here we choose to represent each board position as a list of two integers (its row and column indices, both from 1 to 8).

   A (complete) board configuration of 8 queens is represented as a list of 8 lists, each of which is for the position of one of the 8 queens. For example, the board configuration on the right is represented as

```
******Q*
*Q******
****Q***
*******Q
*****Q**
***Q****
**Q*****
Q*******
```

   ((1 7) (2 2) (3 5) (4 8) (5 6) (6 4) (7 3) (8 1)).

   You are asked to write
   a) a Lisp function which determines if a position on a complete configuration is safe (i.e., is not attacked by any queen already on the board; and
   b) another Lisp function which uses the function defined in a) to test if a complete configuration is a solution of the 8 queen problem.

   Use your program to test if any of the following four board configurations is a solution:
   1) ((1 5) (2 1) (3 4) (3 2) (5 7) (6 3) (7 8) (8 6));
   2) ((1 5) (2 1) (3 4) (4 2) (5 7) (6 7) (7 8) (8 6));
   3) ((1 6) (2 4) (3 7) (4 1) (5 8) (6 2) (7 5) (8 3));
   4) ((1 1) (2 5) (3 2) (4 6) (5 3) (6 7) (7 4) (8 8));
   5) ((1 7) (2 2) (3 5) (4 8) (5 6) (6 4) (7 3) (8 1)).

Submit. You should submit the following:
1. A cover page with your name, class number, date, and a brief summary report of what this project is about. You may also include comments and any points you wish to make in the report.
2. Lisp code you wrote in a text file.
3. The computer printout of the running result of your code with the test data. The output must be in format that can be easily read and understood.

Please zip your submission and email the zipped file to the grader (dorsaz1@umbc.edu) with the subject line "CMSC671 Project1 your first and last name".