Problem Set 8

1. Given an N*N matrix filled with zeros and a single 1 somewhere in it, find row-column of the 1 as fast as you can. You can search N^2 cells of the matrix easily; can you do better?

2. Given two matrices N*N, compute a third matrix that is sum of the two matrices.

3. Given two matrices M*N and N*K, compute a third matrix that is the multiplication of the two matrices.

4. Using the solution to problem 3, compute A^X where A is an input matrix and X is any positive integer.

5. Given two strings (char arrays), check if their concatenation is a palindrome. For instance, *abcdc* and *ba*, *abcdc* and *dcba*, etc. Use string functions from <string.h> rather than developing your own.

6. Check what *grep* utility in linux does. It searches for a string in a file – in a way, similar to what google search does on webpages. For instance, *grep abc ps86.c* prints all the lines in *ps86.c* file which contain the word *abc*. Write a program to read a search string and lines of text from the user, and print only those lines (once) that contain the search string.

7. Implement a two-user tic-tac-toe (use a 3X3 matrix).

8. Implement a one-user tic-tac-toe; the other player is your program.

9. Implement a simple calculator. Read an expression from user, and print its output. For instance, 3 + 5 – 9 should print -1. Improve it to support * and /. Then, add variable assignments. For instance, *a = 12; b = a * 2; b – a* should print *12*.

10. Read dimensions (lengthXheight) of four rectangular walls (d1Xd2, d3Xd4, d5Xd6, d7Xd8) and check if the walls can form a room. For instance, d2 must match d3 and d5 and d7. That is, all the walls must have the same height. Similarly, opposite walls must have the same length.