CSCI 689 **Computer Assignment 5** **Spring 2016**
(30 points)

The Game of Minesweeper

In the game of Minesweeper, a player searches for hidden mines on a rectangular table that might—for a very small board—look like this:

X					X
					X
X	X		X		X
X					
		X			

One way to represent that table in C++ is to use a table of Boolean values marking the mine locations, where true indicates the location of a mine. Given such a table of mine locations: write a C++ program that (1) creates a mine field for a given size; (2) prints out the mine field; (3) creates a table for the mine counts, in which each element indicates the number of mines in the corresponding neighborhood locations that includes the location itself any of the eight adjacent locations that are inside the boundaries of the table; and (4) prints out the table with number of mines corresponding to each location. For the above example, the count table might look like this:

1	1	0	0	2	2
3	3	2	1	4	3
3	3	2	1	3	2
3	4	3	2	2	1
1	2	1	1	0	0
0	1	1	1	0	0

Use two-dimensional vector containers for both the table of a mine field and the table of the mine counts, and in addition to the main ( ) routine, implement the following routines in your program.

- void buildMineField (vector < vector < bool > >& mines, const double& mineProb): This routine determines if a location in the vector of mines has a mine. If that's the case, then sets the value true for that location; otherwise, it sets the value false. The true/false value for each location is randomly chosen with the input argument mineProb, where a random number rnd is generated by the equation: rnd = rand ( ) / (double (RAND_MAX) + 1), and if rnd < mineProb, then it assigns true to that location; otherwise, it assigns false. The random-number generator (RNG) can be initialized by calling the function srand (seed) with the seed value SEED = 1 before generating any random numbers by the RNG.

- void fixCounts (const vector < vector < bool > >& mines, vector < vector < unsigned > >& counts): This routine checks the Boolean value for each location and the values of its eight neighborhood locations in the vector mines, and sets the total count value for that location, which is between 0 and 9, in the vector counts.

- void displayMineLocations (const vector < vector < bool > >& mines): This routine displays the table for the mine field on stdout by printing out the character X for each mine location in the vector mines.

- void displayMineCounts (const vector < vector < unsigned > >& counts): This routine displays the table for the mine counts on stdout for each mine location in the vector counts.

The main ( ) routine gets a pair of numbers from stdin (the size of the mine field and the probability of mine in a location), creates the two-dimensional vectors mineLocations and mine Counts for the size of the mine field, and calls the above routines for each individual task. The process continues until no more input pairs are supplied from the stdin.

If you name your source file as prog5.cc and its header file as prog5.h, then to compile and link your program with the system library routines, first make a link to makefile, in directory: ~cs689/progs/16s/p5, and then execute: make N=5. To test your program, execute: make execute N=5. This command executes your program prog5.exe with the input file prog5.d and displays the output on stdout and also writes in the output file prog5.out. You can find the input and output files of this program in the same directory with the makefile. You need to know that if you name your source and header files different than the suggested names, then the make command won't work.

When your program is ready, mail its source and header files to your TA by executing the following command:

mail_prog prog5.cc prog5.h