CSCE 1030 – Homework 5

Due: 11:59 PM on Tuesday, April 21, 2015 CDT

Problem Statement:

In this C++ program, you will start building the software infrastructure that will allow you to play a simplified version of the class game of Minesweeper that will display to the screen. In Homework 6, you will continue the work begun here to finish the development of the game.

Your program's output should initially call a function to display the department and course number, your name, your EUID, and your e-mail address.

In the `main()` function, your program will prompt the user to enter the number of mines as an integer between 5 and 10, inclusively, that will be placed in a randomly selected location on the 5-by-5 board used for the game. If the user does not enter a valid response, you will repeat this process until a valid response is entered.

To start the game, you will display an introductory message, giving details about the game, including the size of the board, the number of number of mines placed on the board, and the objective of the game (i.e., how many squares are needed to be selected without hitting a mine to win). This will be implemented as a function, passing in parameters as needed.

You shall declare a two-dimensional array in `main()` to represent the 5-by-5 board as an `enum` type you declare to represent the various values that a square can assume. Each square on your board may contain one of the following values (according to the rules of the game):

`*`	initial value of square not revealed (may or may not contain mine)
`@`	a mine is present at this location
`X`	the user selected a square containing a mine
`0 – 8`	the number of mines contained in the immediately adjacent squares

The size of the square board should be declared as a constant, but it is up to you whether or not it is a local or global constant.

You will initialize the board using a function, passing in the two-dimensional array and possibly the size. This function will simply initialize each position on the board to the enumerated type representing the initial value, which is the character `'*'`.

You will then assign the number of mines between 5 and 10, inclusively, specified by the user to randomly generated locations on the board using a `bool` function, passing in the two-dimensional array and possibly the size. This function will generate a random row-column position and then attempt to place a mine at that row-column position on the board. Since the row-column position is randomly generated, if the function attempts to place a mine on a square that already contains a mine, the method will return false, indicating that the assignment was not successful. If the square, however, does not already contain a mine, then the function will assign a mine to that location by assigning

that row-column position with the enumerated type representing the mine and return true, indicative that the assignment was successful. Note that this function attempts to place only one (1) mine at a randomly generated location on the board, so a loop should be implemented in the calling function to achieve this functionality.

Finally, you will display the initial board showing the initial values (i.e., `'*'`) and randomly placed mines (i.e., `'@'`) using a function, passing in the two-dimensional array and possibly the size. This function will display the row and column headers for the board as well as the board itself (see sample output). Note that since your two-dimensional array is an enumerated data type, you will have to map the `enum` value to the representative character that is to be displayed on the board.

You may assume that all input by the user is of the correct data type, though perhaps out of range.

Design:

On a piece of paper (or word processor), write down the algorithm, or sequence of steps, you will use to solve the problem. You may think of this as a "recipe" for someone else to follow. Continue to refine your "recipe" until it is clear and deterministically solves the problem. Be sure to include the steps for prompting for input, performing calculations, and displaying output.

Type these steps into a document (Word, txt, PDF, etc.). Note that this should be done before you start coding as completing it afterwards does not help you in learning the design process.

Implementation:

Now that you have a working design, your next step is to translate these steps into C++ code. Use the algorithm development techniques discussed in class to implement your solution to the problem above. Add your C++ code a little at a time, and compile and test as you go.

Remember to add your comments to your code to explain your program. Do this before/during programming instead of waiting until the end. At a minimum, you should comment the header (e.g., name, class, date, brief description of the program, etc.), all variables (i.e., what they are used for), and specific "blocks" of code. For example, use comments to describe the inputs, the formulas used, and any other important steps, such as loops, in your code.

Your program will be graded based largely upon whether it works correctly on a CSE Department machine, so you should make sure your program compiles and runs on a CSE machine.

Your program will also be graded based upon your programming style. At the very least, your program should include:

- A consistent indentation style as recommended in the textbook and in class;
- Meaningful variable names;
- A block header comment section that includes: your name, e-mail address, and a brief description of the program.

Testing:

Test your program to check that it operates as desired with a variety of inputs to make sure that all "paths" through you code are correct. Sample input and output appears below (with input shown in **bold**):

```
     +---------------------------------------------+
     |       Computer Science and Engineering      |
     |        CSCE 1030 - Computer Science I       |
     |   Student Name     EUID    euid@my.unt.edu  |
     +---------------------------------------------+

                 Welcome to Minesweeper!

Enter number of mines to place on board (5 - 10): 6

------------------------------------------------------------
This computer program will randomly assign  6 mines to the
5 by 5 board. Your objective will be to select ten squares
on the board that do not contain mines using the given in-
formation from the adjacent squares. The game is over when
you either select 10 squares without hitting a mine or you
select a square containing a mine.
------------------------------------------------------------

Initializing board...assigning mines...now let's begin...

    0 1 2 3 4
  +-----------+
 0| * * * @ * |
 1| * * * * * |
 2| @ @ * * * |
 3| * * @ * @ |
 4| @ * * * * |
  +-----------+
```

Documentation:

When you have completed your C++ program, write a short report (2 – 3 paragraphs) describing what the objectives were, what you did to solve the problem, and the status

of the program. Does it work properly for all test cases? Are there any known problems? Also include a reflection in what you learned by completing this program that you anticipate taking forward as your knowledge in C++ programming grows.

Save this report in a separate file to be submitted electronically. You should also include any specific instructions required to compile or execute your code.

Homework Submission:

In this class, we will be using electronic homework submission to make sure that all students hand their programming projects (and labs) on time. You will submit your program source file to the class website through the "**Homework 5**" drop box by the due date and time.

Note that this project must be done individually. The program will be checked using a code plagiarism tool against other solutions, so please ensure that all work submitted is your own.

Note that the dates on your electronic submission will be used to verify that you met the due date above. All homework up to 24 hours late will receive a 50% grade penalty. Later submissions will receive zero credit, so hand in your best effort on the due date.

Summary:

- You will design an algorithm (or steps used) to solve the problem.
- You will implement your program on the CSE machines using C++. You will make sure to use good style, good variable names, indentation, etc. You will compile, run, and test your code.
- You will write a brief report describing what your code does and how well it works.
- You will submit electronically your C++ code, your design, and your brief report.

General Guidelines (for ALL of your programming assignments):

- Your program's output should initially display the department and course number, your name, your EUID, and your e-mail address.
- Use meaningful variable names.
- Use appropriate indentation.
- Use comments, including a program header. Example program header:

```
/*
 ============================================================================
```

```
 Name        : homework2.cpp
 Author      : Mark A. Thompson
 Version     :
 Copyright   : 2015
 Description : The program performs simple arithmetic operations based on in-
               put from the user.
 ============================================================================
 */
```

- Add a header to each function. Example function header:

```
/*
============================================================================
 Function    : deposit
 Parameters  : a double representing account balance and a double represent-
               ing the deposit amount
 Return      : a double representing account balance after the deposit
 Description : This function computes the account balance after a deposit.
============================================================================
 */
```