CSCE 1030 – Homework 5

Due: 11:59 PM on Monday, November 20, 2017

PROGRAM DESCRIPTION

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

REQUIREMENTS

- As with all programs in this course, your program's output should initially display the department and course number, your name, your EUID, and your e-mail address. This functionality will be implemented using a function that you call from your `main()` function.
- You will display an introductory message, giving basic details and rules of the game (see SAMPLE OUTPUT). This functionality will be implemented using a function that you call from your `main()` function.
- You will 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. In addition to being `EMPTY`, that is, unoccupied, each square on your board may contain one of the following values (according to the rules of the game):

`F`	FLAG
`B`	BOMB
`1`	MARSHAL
`2`	GENERAL
`3`	COLONEL
`4`	MAJOR
`5`	CAPTAIN
`6`	LIEUTENTANT
`7`	SERGEANT
`8`	MINER
`S`	SPY

  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. Note that your `enum` type may support additional elements as needed for the implementation.
- You will also declare a corresponding two-dimensional array in `main()` for the 5-by-5 board that represents the two colors, `RED` and `BLUE`, as an `enum` type. This array will keep track of each player's game pieces, either `RED` or `BLUE`, on the

board. Note that your `enum` type may support additional elements as needed for the implementation.

- You will initialize the board using a function, passing in both two-dimensional arrays (i.e., one for the game pieces on the board and the corresponding one for the color of each player's game pieces) and the size. This function will simply initialize each position on the board to the enumerated type representing some initial value, which might be a `' '` on the game board and `NONE` for the color. This functionality will be implemented using a function that you call from your `main()` function.
- You will then create a function, passing in both two-dimensional arrays (i.e., one for the game pieces on the board and the corresponding one for the color of each player's game pieces) and the size, to randomly assign 10 game pieces on each player's back two rows (i.e., rows 0 and 1 for `BLUE` computer and rows 3 and 4 for `RED` player). The game pieces are assigned in the following manner:
  - 1 `FLAG` will be randomly assigned to the back row (i.e., row 0 for `BLUE` and row 4 for `RED`) of any column (i.e., 0 through 4).
  - 3 `BOMB`s will be randomly assigned to one of the two back two rows for each player (i.e., rows 0 and 1 for `BLUE` and rows 3 and 4 for `RED`) and any column (i.e., 0 through 4).
  - 1 of either the `MARSHAL` or `GENERAL`, randomly selected, to by randomly assigned to one of the two back two rows for each player (i.e., rows 0 and 1 for `BLUE` and rows 3 and 4 for `RED`) and any column (i.e., 0 through 4).
  - 1 `MINER` will be randomly assigned one of the two back two rows for each player (i.e., rows 0 and 1 for `BLUE` and rows 3 and 4 for `RED`) and any column (i.e., 0 through 4).
  - 1 `SPY` will be randomly assigned to one of the two back two rows for each player (i.e., rows 0 and 1 for `BLUE` and rows 3 and 4 for `RED`) and any column (i.e., 0 through 4).
  - 3 of any of `COLONEL`, `MAJOR`, `CAPTAIN`, `LIEUTENTANT`, or `SERGEANT`, randomly assigned, to be randomly assigned one of the two back two rows for each player (i.e., rows 0 and 1 for `BLUE` and rows 3 and 4 for `RED`) and any column (i.e., 0 through 4). Duplicates are acceptable.

  Only one game piece may be assigned to an individual position on the board. This means that if the function attempts to place a game piece on a square that already contains a game piece (i.e., not empty), then the function will attempt to place the game piece at another randomly-generated position until it is able to successfully do so. This functionality will be implemented using a function that you call from your `main()` function.

- Finally, you will display the initial board showing each player's 10 game pieces on their back two rows in their respective color (sample code has been provided to implement this functionality) using a function, passing in both two-dimensional arrays and the size. This function will display the row and column headers for the board as well as the board itself (see SAMPLE OUTPUT), where rows are labeled using alphabetic characters A through E and columns are labeled using digit characters 1 through 5. Note that since your two-dimensional array for the game board is an enumerated data type, you may have to cast 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. Please pay attention to the SAMPLE OUTPUT for specific details about the flow and input/output of the program.
- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, course section, date, and brief description), comments for each variable, and commented blocks of code.
- Your program source code should be named "**`homework5.cpp`**", without the quotes.
- Your program will be graded based largely on whether it works correctly on the CSE machines (e.g., `cse01`, `cse02`, …, `cse06`), so you should make sure that your program compiles and runs on a CSE machine.
- You should contact your instructor if there is any question about what is being asked for.
- **This is an individual programming assignment that must be the sole work of the individual student.**

You shall use techniques and concepts discussed in class – you are not to use global variables, `goto` statements, or other items specifically not recommended in this class.

DESIGN (ALGORITHM)

On a piece of paper (or word processor), write down the algorithm, or sequence of steps, that 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.

You should attempt to solve the problem by hand first (using a calculator as needed) to work out what the answer should be for a few sets of inputs. This will also help in designing any loops that will process the two-dimensional array.

Type these steps and calculations into a document (i.e., Word, text, or PDF) that will be submitted along with your source code. Note that if you do any work by hand, images (such as pictures) may be used, but they must be clear and easily readable. This document shall contain <u>both</u> the algorithm and any supporting hand-calculations you used in verifying your results.

SAMPLE OUTPUT (input shown in **bold green**):

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

           W e l c o m e   t o   1 0 3 0   S t r a t e g o

------------------------------------------------------------------------
This program will set up a 5x5 game board for a 1030 version of the game
of Stratego. One player will compete against the computer, each assigned
10 total pieces consisting of the following:
  1 FLAG (F)
  3 BOMB (B)
  1 MARSHAL (1) or GENERAL (2)
  3 COLONEL (3), MAJOR (4), CAPTAIN (5), LIEUTENANT (6), or SERGEANT (7)
  1 MINER (8)
  1 SPY (S)

GENERAL RULES:
--------------
For the most part, the game will follow the standard Stratego rules, al-
though there are some exceptions.
1. Both players (BLUE and RED) will have all of their 10 game pieces as-
   signed randomly with the only requirement being that the FLAG must be
   placed in the back row. RED will start the game first.
2. Higher ranked pieces can capture lower ranked pieces in the following
   order: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> S, meaning that 1 (the
   MARSHAL) can remove 2 (the GENERAL) and so forth. The MINER (8) piece
   may strike a BOMB and remove it to occupy the now unoccupied space. A
   SPY (S), although the lowest ranked piece, may remove the MARSHAL (1)
   or the GENERAL (2).
3. The FLAG and BOMBs are not moveable while all of the other pieces may
   move one square at a time forward, backward, or sideward, but not di-
   agonally to an open square.
4. A player must either move or strike on his/her turn.
5. The game ends when a player strikes his/her opponent's flag.
------------------------------------------------------------------------

Initializing game board...
Assigning BLUE pieces to board...
Assigning RED  pieces to board...
    1 2 3 4 5
  +-----------+
A | 3 B F S 7 |
B | 5 1 8 B B |
C |           |
D | 4 B B 8 3 |
```

CSCE 1030 – Homework 5

Due: 11:59 PM on Monday, November 20, 2017

```
E | S 2 F B 3 |
  +-----------+
$ ./a.out
              +-----------------------------------------------+
              |       Computer Science and Engineering        |
              |        CSCE 1030 - Computer Science I         |
              |   Student Name     EUID    euid@my.unt.edu    |
              +-----------------------------------------------+

           W e l c o m e   t o   1 0 3 0   S t r a t e g o

------------------------------------------------------------------------
This program will set up a 5x5 game board for a 1030 version of the game
of Stratego. One player will compete against the computer, each assigned
10 total pieces consisting of the following:
  1 FLAG (F)
  3 BOMB (B)
  1 MARSHAL (1) or GENERAL (2)
  3 COLONEL (3), MAJOR (4), CAPTAIN (5), LIEUTENANT (6), or SERGEANT (7)
  1 MINER (8)
  1 SPY (S)

GENERAL RULES:
--------------
For the most part, the game will follow the standard Stratego rules, al-
though there are some exceptions.
1. Both players (BLUE and RED) will have all of their 10 game pieces as-
   signed randomly with the only requirement being that the FLAG must be
   placed in the back row. RED will start the game first.
2. Higher ranked pieces can capture lower ranked pieces in the following
   order: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> S, meaning that 1 (the
   MARSHAL) can remove 2 (the GENERAL) and so forth. The MINER (8) piece
   may strike a BOMB and remove it to occupy the now unoccupied space. A
   SPY (S), although the lowest ranked piece, may remove the MARSHAL (1)
   or the GENERAL (2).
3. The FLAG and BOMBs are not moveable while all of the other pieces may
   move one square at a time forward, backward, or sideward, but not di-
   agonally to an open square.
4. A player must either move or strike on his/her turn.
5. The game ends when a player strikes his/her opponent's flag.
------------------------------------------------------------------------

Initializing game board...
Assigning BLUE pieces to board...
Assigning RED  pieces to board...
    1 2 3 4 5
  +-----------+
A | B 2 F 8 7 |
B | S 6 B 4 B |
C |           |
D | B 5 8 B S |
E | 2 5 F 6 B |
  +-----------+
```

SUBMISSION

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

Your program will also be graded based upon your program style. This means that you should use comments (as directed), meaningful variable names, and a consistent indentation style as recommended in the textbook and in class.

- Program Header Example:

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

- Function Header Example:

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

We will be using an electronic homework submission on Blackboard to make sure that all students hand their programming projects on time. You will submit both (1) the program source code file and (2) the algorithm design document to the **Homework 5** dropbox on Blackboard by the due date and time.

Note that this project must be done individually. Program submissions will be checked using a code plagiarism tool against other solutions, including those found on the Internet, 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 and time 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.

As a safety precaution, do not edit your program (using `vim` or `nano`) after you have submitted your program where you might accidentally re-save the program, causing the timestamp on your file to be later than the due date. If you want to look (or work on it) after submitting, make a copy of your submission and work off of that copy. Should there be any issues with your submission, this timestamp on your code on the CSE machines will be used to validate when the program was completed.