Program #7 DUE: On Canvas

Program Purpose

Using Visual Studio 2017 create a CLR console application. Create a program plan and then convert it into C++ statements. Practice debugging, declaring variables, formatting output, if statements and input /output to / from the console window, input validation, loops, functions and 2-D arrays.

Use for your reference

1. Gaddis' book, in-class exercises and your class notes.
2. This assignment sheet & the grade sheet for this lab already printed out.
3. USB Flash drive(s) or other storage media.

LATE PROGRAMS will be not be accepted. Please be sure to upload compressed solution and Word document grade sheet to Canvas by due date/time.

Mandatory Instructions

You will write a program that implements a game of Battleship that will be played between two players. The game should be able to be played numerous times. Each time a game is played one player, the hider, will set up his/her ship locations. Then the other player, the guesser, will attempt to destroy all of the opponent's ships with at most 30 moves. If all ships are destroyed the guessing player wins, if not the hider wins and the guesser loses.

Use functions outlined later to implement this program. Remember that the use of functions greatly reduces the complexity of the program. Stub all functions out and then take time implementing each of them. You are required to implement all the functions. You may add other functions if you wish.

The hider's input will be provided as three values comprised of H=horizontal or V=vertical position of the ship, row, col of the grid. For example if the user enters H 0 0 this would mean ship will have horizontal orientation starting at 0, 0 coordinate on the grid (you will use a 2-D array of type char). Ships should be entered into the array as 'X' characters. Example below shows all 5 ships of various sizes entered into the array. There will always be one ship of size 1, 1 ship of size 2, 1 ship of size 3, 1 ship of size 4 and 1 ship of size 5.

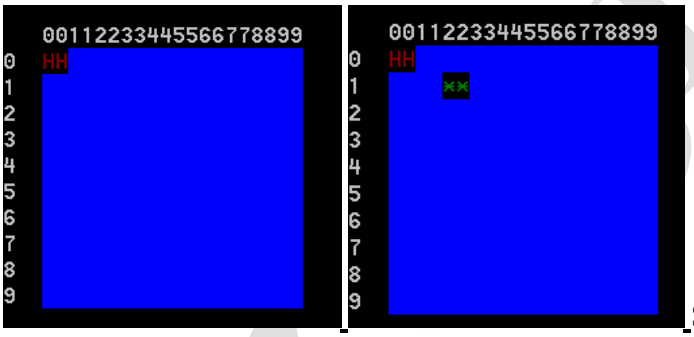
0	1	2	3	4	5	6	7	8	9
X							X		
							X		
							X		
X				X			X		
X				X					
				X					
		X	X	X	X	X			

Program #7

Sample playing of the game:

Game starts with the grid being empty and the hider enters all 5 ships.

Before the game starts, warn the user so hider can switch with the guesser.

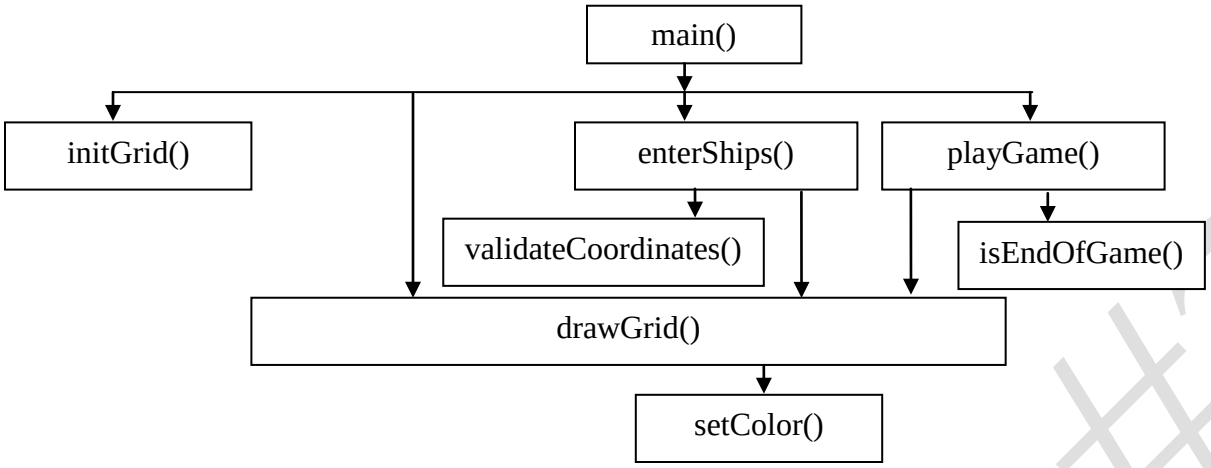


Show progress to the guesser by showing red H for hit, green * for miss.

If the number of guesses exceeds 30, the game ends and guesser loses. After each guess you need to call function `isEndOfGame()` that will return true if the guesser has destroyed all hider's ships (Hint… the grid does not contain any 'X' characters. When guesser hits a ship, place 'H' on the ships grid as well as on the guesses grid. Call the `displayGrid()` function passing whichever grid you need to display.

Program #7 DUE: On Canvas

Hierarchy chart:



Stepwise Refinement:

You have been given a the structure chart which shows the functions that should be part of the program. First create a "stub" program for your solution. A stub program should contain function prototypes, and function headers – all with any needed parameters. Function bodies should be empty with correct return value and braces. The main function should declare variables needed, and the main do while loop that will allow the user to play again. No global variables are allowed. Constants should all be declared globally.

My solutions contain ~ 300 lines of code, comments, white space.

Validating user Input

You need to thoroughly validate user input. Read all input into char variables and then convert into integers when needed. Otherwise display error messages to the user and request re-entry. User may enter X instead of H or V. Allow H or h (V or v) to be valid. Uppercase/lowercase yourself to check what they entered. User may enter 99 for row, col of the coordinate. Do not allow values outside of the valid range (0-9). User may enter 'H' for row or column. Do not accept this as a valid input.

```
==================================================
=  You will enter 5 ships [size 1-5].            =
=  For each ship enter H=horizontal, V=vertical  =
=  followed by row, column coordinates           =
=  Example: H 0 0                                =
==================================================
Ship #1 [1]: 0 0 h
ERROR: enter H or V
Ship #1 [1]: z 0 0
ERROR: enter H or V
```

```
   0011223344556677889 9
0  XX
1
2
3
4
5        XXXX
6
7
8
9
Ship #3 [3]: h 6 5
ERROR: invalid row,col grid coordinate!
Ship #3 [3]:
```

Cannot put next ship in row 6.

```
   0011223344556677889 9
0  XX
1
2
3
4
5        XXXX
6
7
8
9     XXXXXX
Ship #4 [4]: v 9 0
ERROR: invalid row,col grid coordinate!
Ship #4 [4]:
```

Cannot put ship size 4 staring in row 9.

Utilize `isdigit()` and the following in the user input validation:

```
row = atoi(&rowinput);
col = atoi(&colinput);
```

Finally the user may want to enter ships that would be touching each other on the grid. This is now allowed. There must be at least one space between ships. No corners can touch either. Use `validateCoordinate()` function to check for this.

Array Declarations

Declare all your arrays in main. You will need 2 arrays in all. They will be 2D arrays and will represent the hider's grid with the ships and the guessers hit/misses. Both grids are the same size, i.e., 10x10.

char ships[MAX_ROWS][MAX_COLS];
char guesses[MAX_ROWS][MAX_COLS];

The game should be able to be played several times, so you will need to ask the user if he/she wants to play again. You will need to initialize all grids and other variables each time a game is played, i.e., inside your main processing loop.
The following code would initialize array ghips to initial values before a game is played:

```
for (int i=0; i<MAX_ROWS; i++)
      for (int j=0; j<MAX_COLS; j++)
            ships[i][j] = ' ';
```

Function Prototypes

```
void initGrid(char [][MAX_COLS]);
void drawGrid(char [][MAX_COLS]);
void enterShips(char [][MAX_COLS]);
bool playGame(char [][MAX_COLS], char[][MAX_COLS]);
bool validateCoordinates(char, int, int, int, char [][MAX_COLS]);
bool isEOG(char [][MAX_COLS]);
void setColor(int);
```

You may also find the following useful command to clear the console window.

```
system("cls") ;
```

Colors

The game looks much better with color so I will give you some necessary information to get this to work.

Declare constants to hold colors for pen (i.e., font colors) and the background colors.

```
REDPEN=4;                    RED=68;
YELLOWPEN=14;                YELLOW=238;
GREENPEN=2;                  GREEN=34;
BLUEPEN=9;                   BLUE=153;
ORANGEPEN=12;                ORANGE=204;
WHITEPEN=7;
```

You will also need the following code:

1. #include <windows.h>
2. Declare you color constants globally
3. Write a void function setColor taking your constant as a parameter (`colordesired`) and use the statements below to set the desired color for output.

```
HANDLE  hConsole;
hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
SetConsoleTextAttribute(hConsole, colordesired);
```


Program #7

4. Call your setColor function before doing any cout calls.

Program Documentation & Style:

1. Declare all variables and constants that your program uses at the beginning of your program.
2. Your program should include two types of comments:
   Header Comments at the top including lines with:
   - Your name, course name, and class time
   - Program assignment number, program file name (pgm5.cpp) and due date
   - A sentence or two explaining the purpose of the program

   In-line comments: There should be an in-line comment for each main step in your program. In general, this means a comment with each group of C++ statements that handles the input, the processing and the output steps of your program.
3. Use meaningful identifier names
4. Include clear prompts for the user about entering the data.
5. Include clear descriptions of the results when you display them.
6. Each function should have a function documentation header that looks like this:

```
// -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Function:         setColor
// Description:      This function sets foreground/background color for the console window
// Paramters:        int color - color to set
// Returns:          void
// Author:           Jadwiga A. Carlson
// Date:             December 2, 2017
// -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

What to turn in?

1. Logon to CS Classes (Start → Programs → CS Classes). Type in bgsulabs for both the username and password to access CS Classes.
2. Locate and open Carlson → semester →CS2010 →Section X →Lab Y OPEN folder where X is your section number, Y is your lab assignment number and semester is something like sp2012.
3. Drag your ENTIRE application folder into it. Ask for help if you are unsure the first time.
4. You're done.

Program #7