

Programming in C Assignment1

by Matthew Heritage

Submission date: 05-Apr-2019 11:47PM (UTC+0100)

Submission ID: 104253194

File name: 1597033152.Programming_in_C_Assignment1.docx (633.11K)

Word count: 8830

Character count: 41863



Programming in C

Contents

Game Menu	3
Breakdown	3
Functions and variables	3
Flowchart	3
Development.....	3
Test Table	4
Improvements.....	4
Game 1	5
Task breakdown	5
Functions and variables	5
Flowcharts.....	6
Development.....	Error! Bookmark not defined.
Test Table	6
Improvements.....	8
Game 2	9
Task breakdown	9
Functions and variables	9
Flowcharts.....	10
Development.....	Error! Bookmark not defined.
Test table	11
Improvements.....	13
Game 3	14
Task breakdown	14
Functions and variables	15
Flowcharts.....	16
Development.....	Error! Bookmark not defined.
Test Table	16
Appendix	20
Game 1 code	20
Game 2 code	23
Game 3 code	28
References	Error! Bookmark not defined.

Game Menu

Each of the games will have a main menu for them it will have a play button, a help button, and a quit button because each game will use the same menu system below is the breakdown of the game menu. Game 3's menu is modified to include some of the improvements listed below

Breakdown

The menu will be created with 3 options, "1: Play", "2: Help" and "3: Quit" they will be printed to the screen, the program will then ask the user to pick an option between 1 and 3 if they select something out of that range it will ask for a number in range and ask for a number again.

Functions and variables

The main menu will have 5 variables.

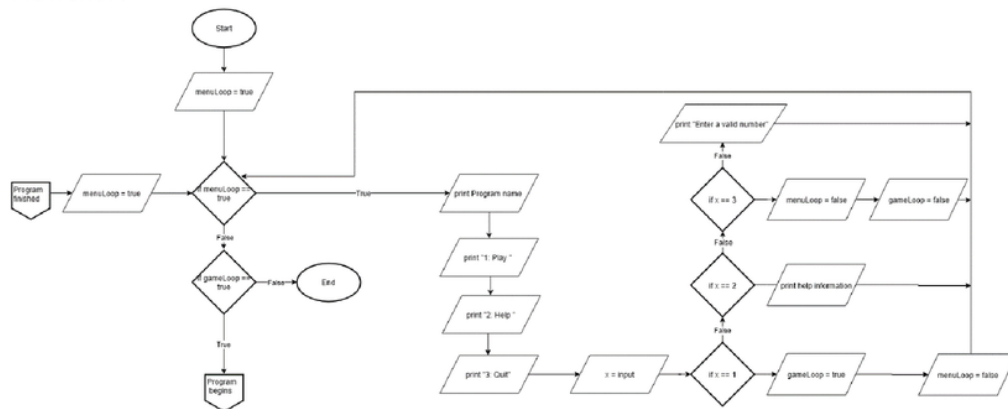
bool menu - will be used in a while loop this will keep the menu looping until the game starts or the menu is exited.

bool help – will be used to display the help for the game.

int h – will be used to hold the player input.

int l,x – Will be used within for loops.

Flowchart



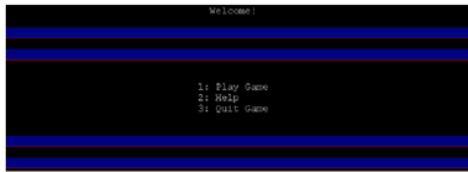
Once the flow reaches "Program begins" the game code is will be after that, and when the game has finished it will return to the menu at "Program finished".

Development

```

Welcome to Guessing game!
1: Play Game
2: Help
3: Quit Game
  
```

The first version of the menu system looked like this this is the version present in game 1 and game 2, game 3 has the improved menu system shown below.



This version has centred the options and has added colour to make it look more like a game.

Test Table

To ensure that the program will function the it is designed to many tests will be carried out it to make sure it runs fine

Test	Test data	Reason	Expected outcome	Actual outcome	Succeed?
1	1	Test if the game will play	Game loads	The game loads	Yes
2	2	Test if the game will display the help	Help prints to the screen	Help prints	Yes
3	3	Test if the game will quit	Game closes	Game closes	Yes
4	7	Enter a number out of range	Enter a valid number	Asked for a valid number	Yes
5	55	Enter a double-digit number out of range	Enter a valid number	Asked for a valid number	Yes
6	a	Not entering a number	Enter a valid number	Repeatedly prints asking for a valid number	No

Improvements

One improvement that could be made to this module would be to make it easier to understand by using different colours to help differentiate from the options because as it currently is it blends together and becomes hard to read, the formatting could also be modified for example the 3 options could be centred like the title to make it easier to follow.

Development

To help with testing all of the games a function was created that would allow a message to be printed in red text a lot quicker and more efficient, this meant to print a message all that would have to be done would be msgCheck("Message here") The string library was included because it would allow to use strings within the function

```
#include "string"
void msgCheck(string j) { SetColour(4); cout << j << endl; SetColour(15); }
```

Another method used when developing the games was this for loop that was used to be able to print all the usable colours within the command line interface to the screen.

```
for (x = 0; x < 255; x++) { SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), x); cout << x; }
```

Game 1

Game 1 is a 2 player number guessing game.

Task breakdown

Below are each of the main tasks to overcome they have been laid out in the order of difficulty with the simplest directly below.

Check what the second player entered

The input the second player enters will be stored in a character they an if statement will check if it is equal to 'h' or 'l' and if anything else was entered it will filter to say "Enter something within range" and it will reduce the loop by one to make sure the current number isn't skipped in the loop.

Checking if player 2 was correct

Once the player 2 input has been filtered to higher or lower, it will then check if the current number in the array is higher or lower than the next number in the array, if true then it will increment their score by 1. Then it will print a message saying if they were correct or wrong.

Taking 7 numbers between 1 and 9 that was not that last number

Allow the first player to enter 7 numbers but when they enter a number it cannot be the same as the last number, this will be done by using an array to store the numbers and when they enter a new number it will check the array for the last index and compare it with the index the player just entered if they are equal it will give the player a message to enter a new number and will remove one from the loop number to redo that number.

Functions and variables

DrawMenu() – Will be used to print the menu to the screen and contain all of the logic for if

CheckNumValid() – This will be used to make sure that any number that the user enters will be valid and that they will be within range.

GuessCheck() – This will be used to check whether players 2 guess is correct or incorrect

ClearBoard() – This will be used to create space on the screen and remove all the text on the screen.

int numbers[] – Will be used to hold all of the numbers that player 1 enters

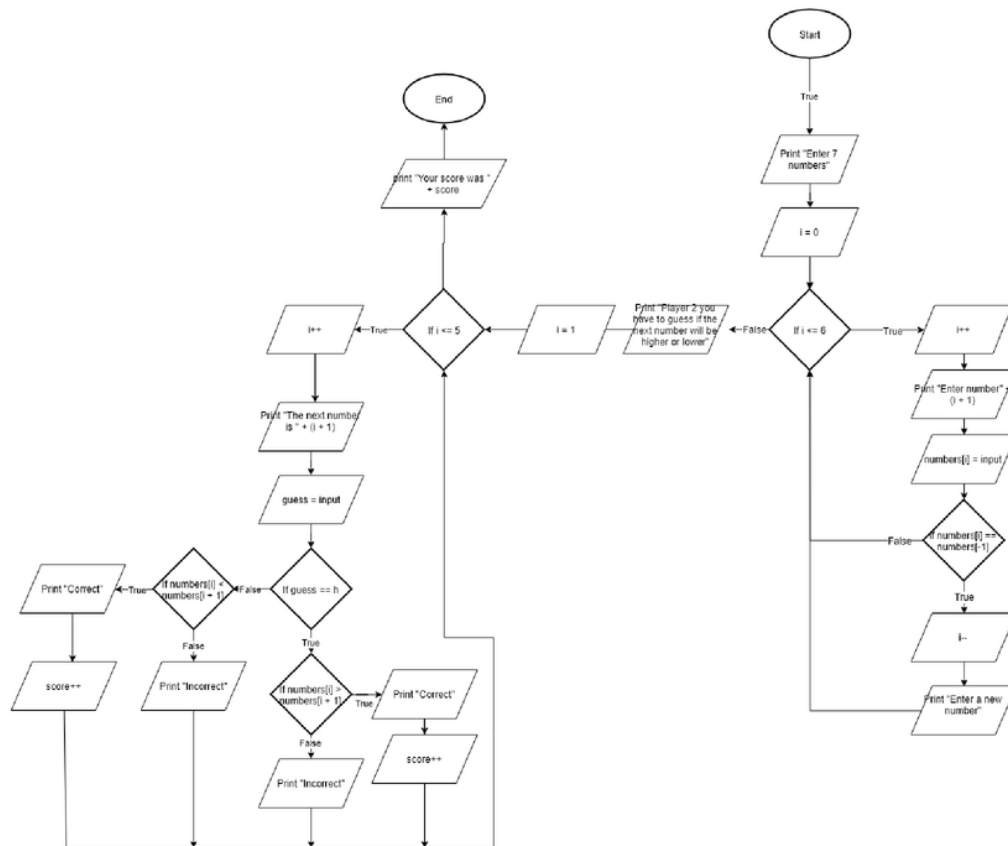
int lastNum – This will be used to hold the last number player 1 entered to stop them entering it again.

char p2g – this will be used to hold the second players guess.

int totalCorrect – Will be used to keep track of the total amount that player 2 has gotten correct.

Flowcharts

Below is a flow chart for the logic of the the game that will be used to program it. The “End” and “Start” for this flowchart connect to the main menu flowchart.



Test Table

To ensure that the program will function it is designed to many tests will be carried out it to make sure it runs fine.

Test	Test data	Reason	Expected outcome	Actual outcome	Succeed?
------	-----------	--------	------------------	----------------	----------

1	N/A	Make sure game plays after menu	Game starts	Enter numbers between 1 and 9, player 1 Enter number 1	Yes
2	1 - 9	Make sure numbers work	Please enter number 2	Enter number 1 1 Enter number 2	Yes
3	0 & 10	Make sure numbers out of range don't work	Please enter a valid number	0 Please enter a valid number	Yes
4	e	Make sure numbers out of range don't work	Please enter a valid number	e Please enter a valid number	Yes
5	H and L	Make sure that H and L work	You are correct/incorrect	h You are CORRECT! The number was 3	Yes
6	G	Make sure another letter cannot be entered	Please enter a valid letter	g Please enter a valid letter	Yes
7	1	Make sure numbers cannot be entered	Please enter a valid letter	1 Please enter a valid letter	Yes
8	N/A	Displays correct score	Score was 5	Congrats you finished the game! your score was 5	Yes
9	N/A	Make sure menu loads after game finishes	Menu loads	The number was 6 Congrats you finished the game! your score was 3 Welcome to Guessing game! 1: Play Game 2: Help 3: Quit Game	Yes
10	3	Make sure the same number cannot be entered twice	Please enter another number	Enter number 1 3 Enter number 2 3 Please enter another number that isnt the last number Enter number 2	Yes
11	N/A	Make sure the help menu displays the correct information	Displays help for game 1	1: Play Game 2: Help 3: Quit Game 3 When the game starts player 1 will enter 7 numbers between 1 and 10. The first number will be displayed for player 2. Player 2 will then have to try and guess if the next number is higher or lower than the first number. This will continue until all of the numbers have been used. Select an item from the menu	Yes

Improvements

To improve this game colours could be used to make the main information stand out more for example the next number could be a different colour to stand out.

Game 2

Game 2 is to make a 2-player treasure hunt game.

Task breakdown

Ask Player 2 for guess

This will use mostly the same as the process for Player 1 for printing the X and Y but when they guess it will be stored in two guess variables.

Draw Board

The board will be printed to the screen, to be able to display numbers at the top it will have a loop that will print out the current loop number to screen so that it will print 1 – 8 along the top. A 2D array is used to be able to hold the contents of the table. To draw the numbers along the side of the board it will follow the same process as the numbers along the top but before it prints the next number it will print out the current row of the array by using a loop with the vertical number as the Y-axis and a new loop for the X-axis, once the X-axis loop has reached 7 the loop will finish and end line for the next row.

Check if the treasure is there

To check if the player guessed was correct or incorrect it will use the array the player 1 entered the locations too and check each index of that array against the 2 guess variables if the answer is correct the location they entered will be used to set an 'X' in the 2D board array, if they entered a wrong answer then it will place an 'N' instead.

The flow chart below is for a function that is used to an X or Y on screen and if the user is making a guess it will store that, and if they are not it will return the next free index in the array placedLoc[] so that it can store the next treasure location in the array. This function is called.

Ask player 1 for X and Y

A loop will be used that only loops twice and the current loop number is used to determine whether it is X or Y and when it is displayed to the screen that loop number will be subtracted from 59 which is the ASCII value for Y so that it will print X or Y out to the screen depending on the loop number. Once the program has displayed the axis it will wait for user input, the input will then be stored in a 1D array.

Stop player from placing in same place

To check that the player cannot place treasure in the same place when the play is asked for coordinates it will check the ones they entered to the array that holds all the coordinates by using a loop that increments by 2 each time it will look in the current location of the loop and the next location to make sure that the same locations have not already been used, if they have then it will remove 1 from the current treasure placing loop and ask them to place it again.

Functions and variables

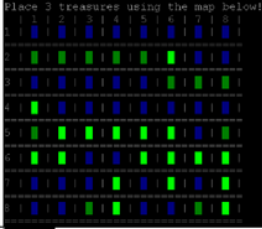
The functions game 2 will use are below

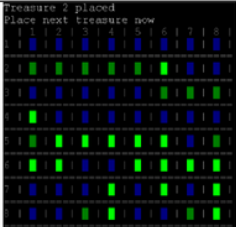
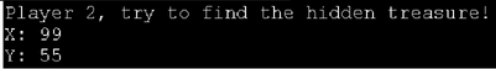
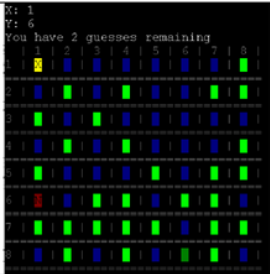
DrawMenu() – This will be used to draw the main menu to the screen.

SetColour() – Be used to change print colour easy.

PlaceGuess() – Used for placing a 'X' or 'N' where player 2 guessed.

5
Test table

Test	Test data	Reason	Expected outcome	Actual outcome	Succeed
1	N/A	Make sure menu loads game	Game loads	Game loads	Yes
2	N/A	Help displays the correct information	Help loads		
3	N/A	Game map loads with background and displays correct	Game board loads		Yes
4	1	Asks user for X and Y:	X:	X:	Yes
5	1-8	Allow number in range	Does not display error message	X: 1 Y:	Yes
6	11 and -1	Make sure number outside of range cannot be entered	Enter a number in range	X: -1 Enter a number within range	Yes
7	a	Enter a character that is not a number	Enter a number in range	X: a Enter a number within range X:	Yes
8	1 And 1	Make sure treasure cannot be stacked on one location	Enter a new number	X: 1 Y: 1 You another location that has not already been used	Yes
9	N/A	Make sure treasure places	Treasure placed	Treasure 2 placed	Yes
10	N/A	Make sure board is reprinted to screen	Board is printed	Board is printed	

11	N/A	Make sure the board colours are the same for that game	It will be the same		Yes
12	N/A	Swap turns once all treasure has been placed	Players 2 turn		Yes
13	55 and 99	Make sure only numbers in range can be entered	Enter a number in range		No
14	A	Make sure other characters cannot be entered	Enter a number in range	Repeatedly shows the help information and game breaks	No
15	1 And 1	Player 2 finding treasure	Will give a message saying they found treasure and colour the treasure location yellow and reset their guesses back to 3		Yes
16	1 And 6	Player 2 not finding treasure	Remove one of their guesses left and place a red marker where they guessed		Yes

1

Matthew Ian Heritage

Student Number: 1810549

17	N/A	Making sure that player 1 can win	Player 1 has won		Yes
18	N/A	Making sure that player 2 can win	Player 2 has won		Yes
19	N/A	Make sure menu loads after game finishes	Menu loads		Yes

Improvements

An improvement for this game would be if player 2 loses it could show the treasure locations after the game is over. A bug that could be fixed would be player can generate guesses by guessing where they already know treasure is to reset their guesses.

Game 3

Game 3 is to make a 2-player noughts and crosses game.

Task breakdown

Menu

For this game the main menu will be improved the 3 options will be centred within the game screen and colour will also be implemented to make it look nicer.

Swap turns

This will be done by just toggling a variable and when the counter is placed it will check what the turn variable is set too and then it will use that character.

Select column to drop

This will be done by printing to tell the user to select a column and then take their user input to store it in a variable it will also make sure that it is between 1 and 9 to stop any out of range numbers from being entered.

Draw board

This will follow the same process from the second game but for this one, it does not need to print the numbers up the side. This board will also use a 1D array instead of a 2D array because it will be easier to code certain tasks and will require one loop less.

Check for win

To check for a win once the dropping counter has finished falling down the counter will check in every position around it and if the next counter is the same it will check the next one after that, until it reaches 4 together of the same counter, but it hits the opposite counter or a default character then it will break and try another combination, it will also check in one direction for 1 counter and if it is the same it will then check the opposite direction for 2 more counters of the same, this will be done to make sure that if a counter is dropped in the middle of counters it will still form a winning condition if there are a total of 4 in a row.

Draw dropping counter

To drop the counters, it will use the column entered by the player draw the counter at the top of the board in the correct column and then check below by looking 10 places further in the array and if it the default character that location will be set to the counter and the position where the counter was will be set back to default. After this the board will be draw again, this process will continue until the piece below is not the default character. This will be done to simulate the counter falling down the board, it will look like someone has dropped a counter from the top of the game down.

Blitz

This will be done by asking the player to select a column and then the one they selected will be used in a loop that will set everything in that column to the default character. It will then set them to the default character.

Time bomb

This will drop using the same system as the normal counters, but once the counter has finished dropping down it will have a loop that will set everything around it to the default character, once this has finished for every direction around it. It will enter a new loop that will go through the entire board making sure that a counter is either at the bottom of the board or that there is a counter below it, if there is not then it will drop that counter down by 1 position this loop will repeat until everything is at the bottom of the board and there are no floating pieces. Once this has finished it will then check every single piece to check if there is a win using the same system as when a piece is dropped.

Functions and variables

SetColour() – Used to change print colour easy.

DrawMenu() – To print the menu.

DrawHelp() – Used to print help.

DrawBoard() – Used to draw game board.

CheckPlacement() – Used to move the counter the player drops to bottom.

ChangePlayerTurn() – Used to toggle between players turns.

PlaceCounter() – Used to place the counter at the top of the board.

CheckWin() – Used to check for a win.

MoveBlitz() – Used to place remove counters in column.

DropCounters() – Used to make sure counters are not floating.

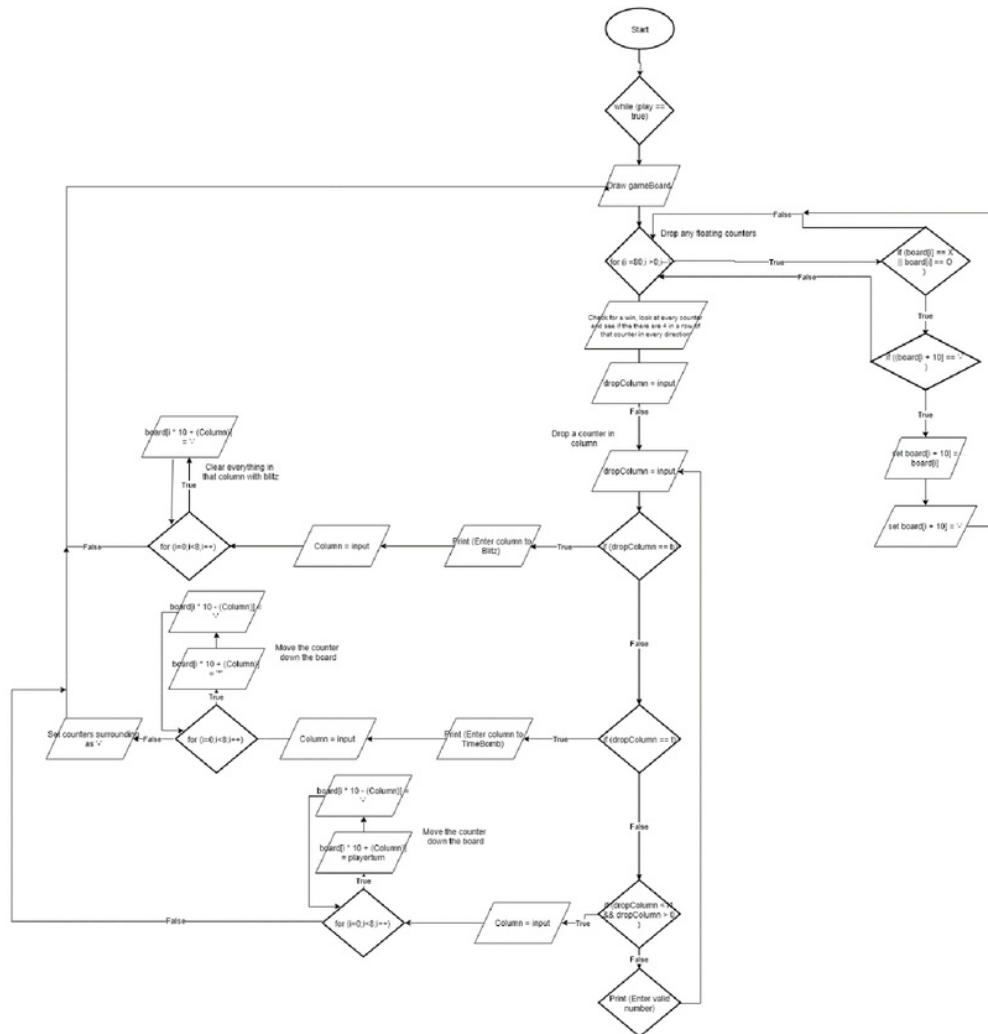
MoveTimeBomb() – Used to drop the Time Bomb to the bottom.

Char board[] – Used to hold all locations on the board.

Int playTurn – Used to keep track on whose turn it is.

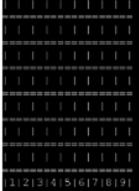
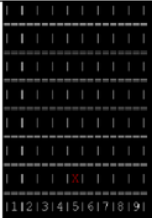
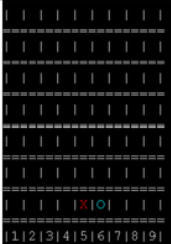
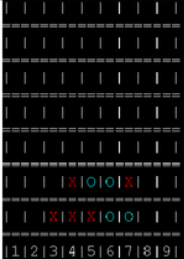
Flowchart

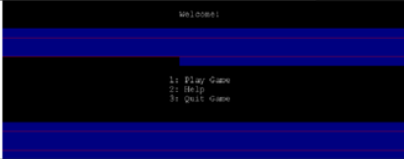
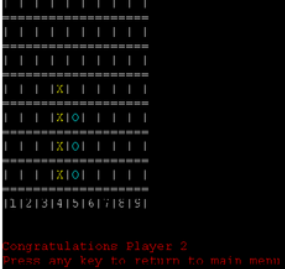
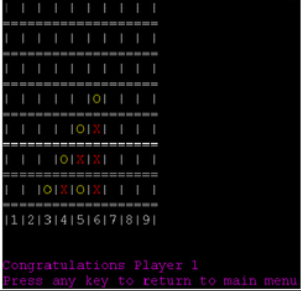
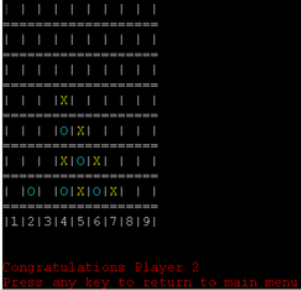
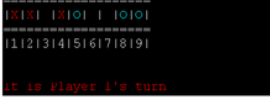
Below is the flowchart for game 3.



Test Table

Test	Test data	Reason	Expected outcome	Actual outcome	Succeed?
1	Running game	Make sure the menu displays correctly	Game menu loads fine		Yes

2	Open help	Make sure the game has the correct help	Game help displays the correct information		Yes
3	Play game	Make sure board loads	Board loads		Yes
4	Play game	Shows whose turn it is	Players 1 turn		Yes
5	5	Place counter in column	X animates dropping to the bottom of board		Yes
6	N/A	Change players turn after turn taken	It is players 2 turn		Yes
7	6	Can place Players 2 counter	O animates dropping to the bottom of board		Yes
8	3-7	Make sure counters can be placed on top of each over	Place counters on top of each over		Yes

9	2-7	Make sure a horizontal win can be detected	Player 2 wins		Yes
10	a	Does menu load after game finishes	Menu loads		Yes, but the menu displays slightly differently
11	4	Make sure that a vertical win can happen	Player 1 wins		Yes, but it displays the wrong player as winner
12	3-6	Check vertical win	Player 2 wins		Yes, but it displays the wrong player as winner
13	2-7	Check vertical win in other direction	Player 1 wins		Yes, but it displays the wrong player as winner
14	1-9	Placing a counter in the middle of 2 on one side and 1 on the other side	Player 1 wins		Yes, but it displays the wrong player as winner

				<pre> ===== X X X X O O O ===== 1 2 3 4 5 6 7 8 9 ===== Congratulations Player 2 </pre>	
15	0	Can number out of range be entered	Please enter a valid number	<pre> It is Player 1's turn 0 It is Player 2's turn </pre>	No
16	B9	Test that blitz can be used to clear a row	Blitz clears the row	<pre> O O O X X X X O 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 </pre>	Yes
17	N/A	Remove ability once used by that player	Will not mention Blitz	<pre> It is Player 2's turn TimeBomb is still available press 'T' to use </pre>	Yes
18	T3	Check that Time Bomb can be dropped like a normal counter and will destroy surrounding counters and can calculate if any counters that dropped can calculate a win	Player 1 win	<pre> X O O O O X X X 1 2 3 4 5 6 7 8 9 It is Player 1's turn Blitz is still available press 'B' to use TimeBomb is still available press 'T' to use X X X X X 1 2 3 4 5 6 7 8 9 Congratulations Player 2 Press any key to return to main menu </pre>	Yes, but it also removed player 2s counter that should have been out of range. It also says the wrong player has won

Appendix

Game 1 code

```
// game1.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream> // Used for cout and cin
#include <windows.h> // Used for increasing the font size

using namespace std;
int CheckNumValid(char inNum) { // Make sure the number is within range

    int num = inNum - 48; // Convert the character from ASCII
    if (num > 0 && num < 11)
    {
        return num;
    }
    else { return -1; cout << "Please enter a number between 1 and 10" << endl; }

}

bool appLoop = true; // Used to make the game loop until exited

int DrawMenu() { // Function used to draw the menu to the screen
    bool menu = true, help = false;
    int i, x;
    char h;
    for (i = 0; i < 24; i++) { // Create space to center the title
        cout << " ";
    }
    cout << "Welcome to Guessing game!" << endl << endl << endl; // Print the
options to the screen
    cout << "1: Play Game" << endl << "2: Help" << endl << "3: Quit Game" << endl;

    while (menu == true) {
        cin >> i;

        if (i == 1) { for (i = 0; i < 30; i++) { cout << endl; } return 1; } //
Play the game
        else if (i == 2) { help = true; } // Help
        else if (i == 3) { appLoop = false; return 3; } // Quit the game
        else { cout << "Enter a valid number" << endl; } // Invalid number
        if (help == true) { // Print help information for the player
            cout << endl;
            cout << "When the game starts player 1 will enter 7 numbers
between 1 and 10." << endl << endl;
            cout << "The first number will be displayed for player 2." <<
endl;
            cout << "Player 2 will then have to try and guess if the next
number is higher or lower than the first number." << endl;
            cout << "This will continue until all of the numbers have been
used." << endl << endl;

            cout << endl << endl << "Select an item from the menu" << endl;
// Return to the main menu loop
            help = false;

        }
    }
}
```

```
        else { cout << "Please enter a valid number" << endl; } // Display
message for invalid number being entered
    }

}

int GuessCorrect(char guess, int curNum, int *numbers) { //Check if player 2 is correct

    bool isHigher;
    if (numbers[curNum - 1] <= numbers[curNum]) //Is the current number bigger than
the last number
    { //Yes
        isHigher = true;

    } else { //No
        isHigher = false;
    }

    if (guess == 'h' || guess == 'H') //Compare what the player entered to the ans
    {
        if (isHigher == true) {
            return 1; //Return 1 if the player gets the ans correct
        }
        else {
            return 0; //Return 0 if the player gets the ans wrong
        }
    }

    else if (guess == 'l' || guess == 'L')
    {
        if (isHigher == false) {
            return 1;
        }
        else {
            return 0;
        }
    }

    else { cout << "Please enter a valid letter" << endl; return 2; } // Catch any
possible errors by returning 2
}

void ClearBoard(int i) { //Clear the screen by creating many clear lines

    for (i = 0; i <= 40; i++) {
        cout << endl;
    }
}

int main()
{
    //Set up font for the application to increase the application size
    CONSOLE_FONT_INFOEX font;
    font.cbSize = sizeof(font);
    font.dwFontSize.Y = 24;
    SetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &font);
    //////////////////////////////////////

    int i, x; //variable for use in the for loop
    int totalCorrect = 0; // Used to store the players score
    char p2g, testNum, lasNum; //the player 2 guess
```

```

int numbers[7]; // define array to hold players 1 numbers

lasNum = -1;

DrawMenu();
while (appLoop == true) {
    cout << "Enter numbers between 1 and 9, player 1" << endl; // start text
    for (i = 0; i <= 6; i++) { // loop to ask for 7 numbers

        cout << "Enter number " << i + 1 << endl;
        cin >> testNum;
        int curNum = testNum;
        char num = numbers[i - 1];
        if (lasNum == curNum) {
            cout << "Please enter another number that isnt the
last number" << endl;

            i--;

        } else {

            numbers[i] = CheckNumValid(testNum);
            if (numbers[i] == -1) { i--; cout << "Please enter
a valid number" << endl; }
        }

        lasNum = testNum;

    }

    ClearBoard(i);

    cout << "Player 2 please enter 'H' or 'L' to guess if the number
is higher or lower." << endl << "The first value is " << numbers[0] << endl;
    for (x = 1; x <= 6; x++) { // main for loop for awnsering
        cin >> p2g;
        int ans = GuessCorrect(p2g, x, numbers);

        if (ans == 0) {
            cout << "You are WRONG!" << endl << "The number was
" << numbers[x] << endl;

        }
        else if (ans == 1) {
            cout << "You are CORRECT!" << endl << "The number
was " << numbers[x] << endl;

            totalCorrect++; // Add 1 to the player score
        }

        else { x--; } // Make sure the number isnt skipped
        if (x == 6) { break; }
    }
    cout << endl;
    cout << "Congrats you finished the game! your score was " <<
totalCorrect << endl << endl;
    DrawMenu(); // Draw menu after game done

    ClearBoard(i);

}

return 0;
}

```

Game 2 code

```

// game2.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream>
#include <windows.h> // Used for changing the colour of the CML
#include "string" // Used for testing
#include <ctime> // Used to be able to use the sleep function
using namespace std;

char board[8][8]; //Used to store all the the coords on the board, has one index
bigger to fix bug

int colourCoords[100]; // The storage for the colours of each location
bool appLoop = true; //Used in loop to keep the program running
int lives = 3; // Used to store the amount of lives the player currently has
int GX, GY; //Used for the player 2's guess
int correctG; // Used to store the total amount of correct guesses
int treasuresFound = 0; // Used to track how many Treasures have been found in game

void SetColour(int colour) { // A function used to be able to change the colour of
text in a simplier method
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), colour); // Set the
colour of all printed text
}

int DrawMenu() { // Used to draw the main menu to the screen
    bool menu = true, help = false; //Menu used for a loop to keep menu cycling if
a wrong input was entered or if help was loaded
    int i, x;
    char h;
    for (i = 0; i < 24; i++) { // Create space to center the title
        cout << " ";
    }
    cout << "Welcome to Treasure Hunt!" << endl << endl << endl;
    cout << "1: Play Game" << endl << "2: Help" << endl << "3: Quit Game" << endl;
    // Display options for menu

    while (menu == true) {
        cin >> i;
        i = int(i);
        if (i == 1) { for (i = 0; i < 30; i++) { cout << endl; } return 1; } //
Play game
        else if (i == 2) { help = true; } // Show help
        else if (i == 3) { appLoop = false; return 3; } // Quit game
        else { cout << "Enter a valid number" << endl; } // Catch any
numbers out of range
        if (help == true) { // Display help
            cout << endl;
            cout << "When the game starts a randomised area of land and ocean
will show up." << endl << endl;
            cout << "Player 1 will enter 3 locations on where to hide
treasure on the map." << endl;
            cout << "Player 2 will have to try and guess where player 1 has
hidden the treasure." << endl;
            cout << "They have 3 attempts to find one, if they find one in
them attempts then they" << endl << "will get 3 lives again " << endl << endl;
            SetColour(14); // Change text colour to yellow
            cout << endl << endl << "Select an item from the menu" << endl;

```

```

        help = false;
        SetColour(15); // Change text colour back to white
    }
    else { cout << "Please enter a valid number" << endl; i = 2; } // Catch
any out of range numbers
    }

}

void PlaceGuess(int GuessX, int GuessY, int *placedLoc) { // Used for when the player
2 has entered a guess on where they think the treasure is
    int x;
    bool i = false;
    lives--; // Remove a life because if they get it correct their lives will get
reset back to 3
    for (x = 0; x <= 5; x = x + 2) { // Check the location where the player 2
entered
        if (GX == placedLoc[x]) {
            if (GY == placedLoc[x + 1]) { // If treasure is found
                SetColour(14); cout << "You found a treasure chest!" <<
endl;

                board[GY][GX] = 'X';
                treasuresFound++;
                lives = 3; // Set lives back to the default
                i = false;
                break;
            }
            else { board[GY][GX] = 'N'; } // Treasure not found
            break;
        }
        else { board[GY][GX] = 'N'; } // Treasure not found
    }

}

}

// Make sure players cannot place in the same location as where there is already
treasure too

void DrawBoard() { // Used to draw the game board to the screen
    int x, i = 0, y, side = 1, z;
    SetColour(8); //Grey
    cout << " | ";
    for (y = 1; y <= 8; y++) { //Draw numbers on top

        cout << y;
        SetColour(8);
        cout << " | ";
    }
    cout << endl;
    for (y = 0; y < 8; y++) {
        //Create a loop that will just divide the first 10 nums of colourCoords
down to normal nums

        cout << side; side++;
        SetColour(8);
        cout << " | ";

        for (x = 0; x < 8; x++) { //Place content of the board
            //Used to set the background colours
            if (colourCoords[i] > 150) { SetColour(17); } //Set Blue (Water)
            else if (colourCoords[i] < 30) { SetColour(34); } //Set dark
Green

```

```
        else { SetColour(170); } //Set light Green
        i++;
        if (board[y][x] == 'X') { SetColour(224); } else if (board[y][x]
== 'N') { SetColour(64); } // Enter the contents of the board
        cout << board[y][x];
        SetColour(8);
        cout << " | ";

    }
    cout << endl;
    for (x = 0; x < 35; x++) { //Create spacing between the rows
        cout << " ";
    }
    cout << endl;
}
SetColour(15);
}

int PlaceTreasure(int coor, bool isGuess, int *placeLoc) { // Used to draw the X and Y
to the screen
    int i;

    cout << char(89 + coor) << ": "; // Convert the axis to ascii to work out what
symbol to display
    if (isGuess == false) { //Used for player 1
        7
        for (i = 0; i <= 5; i++) {
            if (placeLoc[i] == -1) { break; } // This makes 'i' = to the next
empty array loc
        }

        return i;
    } else // Used for player 2 when they are guessing
    {
        if ((89 + coor) == 88) { // This is for X
            cin >> GX; GX = GX - 1;
        }
        else if ((89 + coor) == 89) //This is for Y
        {
            cin >> GY; GY = GY - 1;
        }
    }

    return 0;
}

void SetupGame() { // Set all the default the game will use
    int i, x, y, lake = 0;
    for (y = 0; y <= 100; y++) { //Create random landscape
        colourCoords[y] = (rand() / 100) + lake;
        lake = 0;
        if (colourCoords[y] < 30) { lake = -140; } // If last loc was water then
increase the chance of next being water
    } // This is done to make it less likely for just one location to be water and
more likely for a body of water
    24
    for (i = 0; i <= 10; i++) {
        for (x = 0; x <= 10; x++) { // Set the default board character
            board[i][x] = '#';
        }
    }
}
```

```
    }
    lives = 3;

}

void ResetBoard() { //Created to make sure that the board can only contain the
specified chars due to a bug that was appearing
    int x, y;
    for (x = 0; x < 7; x++) {
        for (y = 0; y < 7; y++)
        {
            if (board[x][y] != '#' && board[x][y] != 'X' && board[x][y] !=
'N')
            {
                board[x][y] = '#';
            }
        }
    }
}

int main()
{
    //Increase font size to make the application bigger
    CONSOLE_FONT_INFOEX font;
    font.cbSize = sizeof(font);
    font.dwFontSize.Y = 24;
    SetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &font);
    //
    srand(time(NULL)); //Set seed for random

    bool gameLoop, usedNum = false;
    int x, y, i, c;
    char input; //used to hold the input to validate it
    int PX = 0, PY = 0;

    int placedLoc[10]; // Used to hold the locations that the treasure is stored in
    while (appLoop == true) { //Keep application running until exited
    {

        gameLoop = false;

        if (DrawMenu() != 3) { gameLoop = true; } //Draw menu

        if (gameLoop == true) // Loop to keep the game playing
        {
            SetupGame(); // Sets all vars to default
            input = 'q';
            for (x = 0; x <= 6; x++) { placedLoc[x] = -1; } // Clear
the treasure locations

            cout << "Place 3 treasures using the map below!" << endl;
```

```

        DrawBoard();
        for (x = 1; x <= 3; x++) { //Place the treasure in locs
            for (y = -1; y <= 0; y++) { // Toggle between X and
Y
                i = PlaceTreasure(y, false, placedLoc); //
                Print x or y and answer in i
                cin >> input;
                if (int(input) > 48 && int(input) < 57) //
                Make sure that only numbers can be entered
                {
                    placedLoc[i] = input - 49; //Convert
                the number from ASCII
                if (PX == 0) { PX = input - 49; } else
                { PY = input - 49; } // Set the current coords
                }
                else { cout << "Enter a number within range"
                << endl; y--; } //Out of range catch
            }

            for (c = 0; c <= 6; c = c + 1) {
                if (placedLoc[c] + 1 == (PX + 1) &&
                if (usedNum == false) { usedNum
                = true; }
                else { // Stop player 1 from
                entering the same location twice
                SetColour(4); cout <<
                "Use another location that has not already been used" << endl; SetColour(15);
                if (x < 1) { x = 1; }
                else { x--; }

                Sleep(1250); //delay for
                1.25 seconds
            }
        }

        }
        usedNum = false;
        PX = 0; PY = 0;
        cout << endl << endl << endl << endl << endl;

        cout << "Treasure " << x << " placed" << endl;
        cout << "Place next treasure now" << endl;
        DrawBoard(); // Draw board after each treasure is
        placed
    }

    for (x = 0; x < 30; x++) { //Create space so that player 2
    cannot see what player 1 entered
        cout << endl;
    }
    cout << "You have " << lives << " guesses remaining" <<
    endl; // Print out the users guesses remaining
    DrawBoard();

```

```

        cout << endl << "Player 2, try to find the hidden
treasure!" << endl;
        while (0 <= lives - 1) {
            for (y = -1; y <= 0; y++) { // Alternate between
axis
                PlaceTreasure(y, true, placedLoc);
            }
            PlaceGuess(GX, GY, placedLoc); //Check if what
player 2 entered was a correct coordinate
            ResetBoard(); // Reset all the of the characters on
the board
            cout << "You have " << lives << " guesses
remaining" << endl;
            DrawBoard();

            if (treasuresFound >= 3) { cout << "Player 2 has
found all of the treasure and has won!" << endl << "Please wait..." << endl;
Sleep(4000); break; } //Player 2 win and wait 4 seconds
        }
        if (treasuresFound < 3) { cout << "Player 2 failed to find
all of the treasure, Player 1 has won" << endl << "Please wait..." << endl;
Sleep(4000); } // player 1 win and wait 4 seconds
    }
    for (y = 0; y < 25; y++) { cout << endl; } //Create space
after the game

    }

}

return 0;
}

```

Game 3 code

```

// game3.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream>
#include <windows.h>
using namespace std;

int test, bColumn = -1;
char board[80]; // Contains the contents of the array
bool p1Turn = false, p1HUA[3], p2HUA[3]; //Player has used ability
bool gameLoop = true; // Make sure the gameplay logic keeps looping
int playTurn = 1; //Alternative variable to track players turn because of error that
was occurring
int winLocs[2]; //Used to check for wins

void SetColour(int colour) { // Function to make changing print colour easy
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), colour); //Changes the
text colour
}

void DrawMenu() { // Function to draw the main menu for the game
    int i, option;

```

```

SetColour(0); 23for (i = 0; i < 35; i++) { cout << "-"; } SetColour(15);
cout << "Welcome!" << endl;
cout << endl; SetColour(28);
for (i = 0; i < 80; i++) { cout << "_"; } cout << endl;
for (i = 0; i < 80; i++) { cout << "-"; }
SetColour(15);
for (i = 0; i < 35; i++) { cout << "-"; }

cout << endl << endl;
for (i = 0; i < 33; i++) { cout << " "; }
cout << "1: Play Game" << endl;

for (i = 0; i < 33; i++) { cout << " "; }
cout << "2: Help" << endl;

for (i = 0; i < 33; i++) { cout << " "; }
cout << "3: Quit Game" << endl;
cout << endl << endl; SetColour(28);
for (i = 0; i < 80; i++) { cout << "-"; } cout << endl; //Draw lines to add
colour to menu
for (i = 0; i < 80; i++) { cout << "-"; }
SetColour(15);
}

void DrawHelp() { //Function to draw the help for the game
    int x;
    char i;
    for (x = 0; x < 33; x++) { cout << endl; }
    for (x = 0; x < 33; x++) { cout << " "; }
    cout << "HOW TO PLAY!" << endl << endl;
    cout << "To play you have to enter the column number you want to place the
counter in." << endl;
    cout << "Once this has been entered you will see your counter fall in to the
specified location" << endl << endl;
    cout << "Each player has two abilities they can use once" << endl << endl;
    SetColour(11); cout << "The first one is the Blitz it can be used to remove all
counters from a column" << endl << endl;
    SetColour(5); cout << "The second one is the TimeBomb it drops like a normal
counter but once it lands it will destroy all the surrounding counters and any
counters on top will drop down" << endl << endl; SetColour(14);
    cout << endl << endl << "Enter any key to return to the main menu" << endl <<
endl;
    SetColour(15);
    cin >> i;
    for (x = 0; x < 33; x++) { cout << endl; }
}

void DrawBoard(char *board, int colour) { // Used to draw the game board

    int x, y, loopNum, mult = 0, i, alt;
    loopNum = 0;

    alt = 15;

    for (y=0;y<30;y++)//Make space before drawing the board
    {
        cout << endl;
    }
7for (y = 0; y < 7; y++) { //Vertical loop for board

        cout << "|";
        for (x = 0; x < 9; x++) { // Horizontal loop for board

```

```

        if (board[x + (loopNum * 10)] == 'X') { // Change colours for
counters
            SetColour(12); // Red
        }
        if (board[x + (loopNum * 10)] == 'O') {
            SetColour(11); // Light blue
        }
        if (board[x + (loopNum * 10)] == '*') {
            SetColour(13); // Pink
        }

        if (board[x + (loopNum * 10)] == '-') { // Hide all the empty
slots in the board to make it easier to look at
            SetColour(0); // Black
        }
        for (i = 0; i <= 2; i++) {
            if (board[x + (loopNum * 10)] == winLocs[i]) { // If win
then change colour of winner counters
                SetColour(14); // Yellow
            }
        }
        cout << board[x + (loopNum * 10)]; //This prints the
content of the table

        SetColour(15); // White
        cout << "|";

    }
    cout << endl;
    for (x = 0; x < 19; x++) { // Create lines between each position
        SetColour(15);
        cout << "=";
    }
    loopNum++;
    cout << endl;
}
cout << "|"; //Add one more for the start of the nums
for (x = 0; x < 9; x++) { //Add the numbers at the bottom

    cout << x+1 << "|";

}
for (x = 0; x < 3; x++) { //Make space after the board
    cout << endl;
}

}

```

```

int CheckPlacement(int loc, char *board) { // Move counters down if they are
floating
    int x;

    for (x = 0; x <= 6; x++) {

        if (board[loc + 10] == '-') // Check position below
        {

```

```
        loc + 10;
        test = (x + 1);

    }

    else
    {
        return loc - 20;
    } //return the current location in array to know the next filled
place{

    }

    return 70; //Return 70 to define nothing in the row
    //This will drop the counter to the bottom

}

void ChangePlayerTurn() { // Toggle player turns
    //Display player information such as player turn and what abilitys are still
    available
    if (playTurn == 1) { p1Turn = false; SetColour(11); cout << "It is Player 2's
    turn" << endl; playTurn = 2; SetColour(8);

        if (p2HUA[2] == false) { cout << "Blitz is still availble press 'B' to use" <<
    endl; }
        if (p2HUA[1] == false) { cout << "TimeBomb is still availble press 'T' to use"
    << endl; }

    }
    else { p1Turn = true; SetColour(12); cout << "It is Player 1's turn" << endl;
    playTurn = 1; SetColour(8);

        if (p1HUA[2] == false) { cout << "Blitz is still availble press 'B' to use" <<
    endl; }
        if (p1HUA[1] == false) { cout << "TimeBomb is still availble press 'T' to use"
    << endl; }

    }

}

int yLoc; // Location of the just placed counter
bool blitz = false; //Is current counter a TIMEBOMB not BLITZ
bool PlaceCounter(int loc, int firstLoc, int loopNum) { // Places counter at top and
will drop down

    if (board[(firstLoc + (loopNum * 10)) ] == '-') { //Check if down is empty

        int currentY = firstLoc + (loopNum * 10);
        if (currentY >= loc) { return false; }
        board[currentY - 10] = '-'; //Remove last place
        if (blitz == true) {
            board[currentY] = '*'; //Place in current place
            yLoc = currentY;

        }
        else if (playTurn == 1) {
            board[currentY] = 'X'; //Place in current place
            yLoc = currentY;

        }

    }

}
```

```

        else if (playTurn == 2) {
            board[currentY] = 'O'; //Place in current place
            yLoc = currentY;
        }
    }
    else { return false; }

    return true;
} int CheckLoc(int counterNo, int finalLoc, char turn, int look) {
    //counterNo is the amount of Counters in a row
    //finalLoc is where the counter ended
    //Turn is to get whose turn it is to know what counter to check
    //Look is the number passed to see what directions to look in
    //row
    int x;
    counterNo = 0;
    for (x = 0; x <= 3; x++) { //Look one way +
        if (board[finalLoc + (x * look)] == turn) { // check the next 3 places
            winLocs[x] = board[finalLoc + (x * look)];
            counterNo = counterNo + 1; // Add 1 to the current counters in a
        }
    }

    if (counterNo >= 4) { return 1; } // return 1 for a win
    else if (counterNo == 3) { // Check the counter on other side
        if (board[finalLoc - look] == turn) {
            winLocs[3] = board[finalLoc - look];
            counterNo = counterNo + 5;
            if (counterNo >= 4) { return 1; } // return 1 for a win
        }
    }

    counterNo = 0; // reset the counter
    for (x = 0; x <= 3; x++) { //Look one way -
        if (board[finalLoc - (x * look)] == turn) { // check the next 3 places
            winLocs[x] = board[finalLoc - (x * look)];
            counterNo = counterNo + 1;
        }
    }

    if (counterNo >= 4) { return 1; } // return 1 for a win
    else if (counterNo == 3) { // Check the counter on other side
        if (board[finalLoc + look] == turn) {
            winLocs[3] = board[finalLoc + look];
            counterNo = counterNo + 5;
            if (counterNo >= 4) { return 1; }
        }
    }

    counterNo = 1; // reset the counter

    return 0; // return 0 for nothing
}

void clearWinLoc() {
    int i;
    for (i = 0; i < 3; i++) {

```

```

        winLocs[i] = -10;
    }

}

int CheckWin(int finalLoc) { // Check for any win conditions
    int counterNo=0;
    bool win = false;
    char turn;

    if (p1Turn == true) { turn = 'X'; } else { turn = 'O'; }

    if (CheckLoc(counterNo, finalLoc, turn, 10) == 1) { cout << "WIN VERTICALLY ";
gameLoop = false; win = true; } //Look vertically

    if (CheckLoc(counterNo, finalLoc, turn, 1) == 1) { cout << "WIN HORIZONTALLY ";
gameLoop = false; win = true; } //Look horizontal

    if (CheckLoc(counterNo, finalLoc, turn, 11) == 1) { cout << "WIN 11 DIAGONALLY ";
gameLoop = false; win = true; } //Look up right diagonal / down left diagonal

    if (CheckLoc(counterNo, finalLoc, turn, 9) == 1) { cout << "WIN 9 DIAGONALLY ";
gameLoop = false; win = true; } //Look up left diagonal / down right diagonal

    if (win == false) { clearWinLoc(); }
    return 1;
}

void MoveBlitz() { // Clear a column with blitz
    int column= -1, i;

    20
    cout << "Please enter the column number you would like to blitz" << endl;
    cin >> column;
    column--;
    bColumn = column;
    if (column > 0 && column < 10) {
        for (i = 0; i < 7; i++) {
            board[column + (i*10)] = '-';
            ChangePlayerTurn();
            DrawBoard(board,38);
        } // Clear everything in that column and set it back to default
character
    }

}

int blitzLandLoc = -1;
void DropCounters(int check) { //Function to drop any floating counters
    if (blitzLandLoc != -1) {
        DrawBoard(board, 15);
        Sleep(150);
        int x, i;
        for (i = 0; i < 5; i++) {
            for (x = 71; x > 0; x--) {

                if (board[x] == '-') {

```

```

        if (board[x - 10] == 'X') { // Check if any there
are any floating X counters
            board[x] = 'X'; // Drop down a row
            board[x - 10] = '-'; // Clear the position it
was in
        }
        else if (board[x - 10] == 'O') { // Check if any
there are any floating O counters
            board[x] = 'O'; // Drop down a row
            board[x - 10] = '-'; // Clear the position it
was in
        }
    }

    }

    DrawBoard(board, 15);
}
19 for (i = 0; i < 70; i++) { // Check for a win in every position after
every counter has finished dropping
    if (board[i] == 'X' || board[i] == 'O')
        CheckWin(i);
    }
}

}

int MoveTime() { // Place time bomb on board
    int column;

    cout << "Select a column to place TimeBomb" << endl;
    cin >> column;
    column--;
    blitz = true;
    return column;
}

void ExplodeTime(int finalLoc) { // Explode time bomb
    int i;
    for (i = 0; i < 70; i++) {
        if (board[i] == '*') {
            // Set the surrounding spaces back to default
            blitzLandLoc = i;
            board[i] = '-';
            board[i + 1] = '-';
            board[i - 1] = '-';
            board[i + 9] = '-';
            board[i - 9] = '-';
            board[i + 10] = '-';
            board[i - 10] = '-';
            board[i + 11] = '-';
            board[i - 11] = '-';
            DropCounters(board[(i - 11) - 10]);
            CheckWin(i); // Drop all the floating counters and check for a
win
        }
    }
}

```

```

        DrawBoard(board, 15);

        blitzLandLoc = -1; // Reset Blitz for next use
    }
}

int main()
{
    //Increase font size to make the application bigger
    CONSOLE_FONT_INFOEX font;
    font.cbSize = sizeof(font);
    font.dwFontSize.Y = 24;
    SetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &font);
    //

    p1HUA[2] = false; p1HUA[1] = false; // P1s abilities
    p2HUA[2] = false; p2HUA[1] = false; // P2s abilities

    bool play = false; // Loop for gameplay
    int x = 0, loc, placeLoc;
    char playerInput; //Drop location
    char inputToQuit;

    DrawMenu();

    while (x != 3 || x != 1) { // Main menu
        cin >> x;
        if (x == 1) { play = true; gameLoop = true; break; } //Play game
        else if (x == 3) { play = false; gameLoop = false; break; } //Quit game
        else if (x == 2) { DrawHelp(); DrawMenu(); play = true; gameLoop =
true; } //Help
        else { x = -1; cout << "Please enter a valid number!"; } // Invalid
number
    }
    if (x == 1) { play = true; gameLoop = true; } // Play game
    else if (x == 2) { play = false; gameLoop = false; } // Quit game

    while (play == true) {

        for (x = 0; x < 80; x++) { board[x] = '-'; } // Setup the default values
for the board
        DrawBoard(board,15); //Draw the board to the screen

        playTurn = 1; // Set turn to players 1 turn
        SetColour(12); // Red
        cout << "It is Player 1's turn" << endl;
        while (gameLoop == true) { // Loop that keeps the game running

            SetColour(15); //White
            cin >> playerInput;

            if (playerInput == 'b') { //Blitz
                //Checking the player hasnt already used this ability,
                Player turn is being called because it will get called again in main
                if (playTurn == 1) { if (p1HUA[2] == true) { cout << "You
have already used this ability" << endl; return -2; } }

```

```

        else { if (p2HUA[2] == true) { cout << "You have already
used this ability" << endl; return -2; } }

        //Set it so that player has used their ability

        if (playTurn == 1) { p1HUA[2] = true; }
        else { p2HUA[2] = true; }
        MoveBlitz(); } //Use blitz to clear column

    else if (playerInput == 't') { //Time bomb

        //Checking the player hasnt already used this ability,
        Player turn is being called because it will get called again in main
        if (playTurn == 1) { if (p1HUA[1] == true) { cout << "You
have already used this ability" << endl; return -2; } }
        else { if (p2HUA[1] == true) { cout << "You have already
used this ability" << endl; return -2; } }

        //Set it so that player has used their ability
        if (playTurn == 1) { p1HUA[1] = true; }
        else { p2HUA[1] = true; }

        int a;
        a = MoveTime();
        loc = CheckPlacement(a, board);

        for (x = 0; x < test; x++) { //Move time bomb down the
board

            if (PlaceCounter(loc, a , x) == false) { break; }
            DrawBoard(board, 15);

            Sleep(200); // Delay 200 milliseconds to fake an
animation

        }
        blitz = false;

        ExplodeTime(loc); // Destroy surrounding counters
        CheckWin(yLoc); //Check for a win
        ChangePlayerTurn(); //Toggle player turn
    }

    else { // Drop normal counter
        placeLoc = playerInput - 48;
        placeLoc = placeLoc - 1;
        loc = CheckPlacement(placeLoc, board);

        for (x = 0; x < test; x++) { // Drop the counter down the
board

            if (PlaceCounter(loc, placeLoc, x) == false)
{ break; }

            DrawBoard(board,15);
            Sleep(200); // Delay 200 milliseconds to fake an
animation

        }

        CheckWin(yLoc); //Check for win in column counter was
placed in

        ChangePlayerTurn(); //Toggle turn
    }
}

```

1

Matthew Ian Heritage

Student Number: 1810549

```
        DrawBoard(board,15); // Redraw the board once counter has finished
dropping
        if (playTurn == 1) {SetColour(13); cout << "Congratulations Player " <<
playTurn << endl; //Check if there has been a win
        } else { SetColour(12); cout << "Congratulations Player " << playTurn <<
endl; } //Check if there has been a win

        cout << "Press any key to return to main menu" << endl; // Allows player
to see the board before game returns to main menu
        cin >> playerInput;
        for (x = 0; x < 30; x++) { cout << endl; } //Creates space
        DrawMenu();

        while (x != 3 || x != 1) { //Draws main menu
            cin >> x;
            if (x == 1) { play = true; gameLoop = true; break;}
            else if (x == 3) { play = false; gameLoop = false; break; }
            else if (x == 2) { DrawHelp(); DrawMenu(); play = true; gameLoop
= true; }
            else if (x == -1) {} //Used to stop program from spamming "Please
enter valid number"
            else { cout << "Please enter a valid number! "; x = 5; }
        }

        return 0;
}
```

References

References

- [1] Change the print colour [Online]{<https://docs.microsoft.com/en-us/windows/console/setconsoletextattribute>}[Accessed on 04/03/19]
- [2] Change the font size to create a bigger window [Online]{ <https://docs.microsoft.com/en-us/windows/console/setcurrentconsolefontex>}[Accessed on 05/03/19]
- [3] C++ logo [Online] {<https://isocpp.org/>} [Accessed on 04/04/19]

Programming in C Assignment1

ORIGINALITY REPORT

8%

SIMILARITY INDEX

4%

INTERNET SOURCES

1%

PUBLICATIONS

7%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to Pembrokeshire College

Student Paper

3%

2

Submitted to University of Wales, Lampeter

Student Paper

1%

3

Submitted to University of Southern California

Student Paper

<1%

4

www.dreamincode.net

Internet Source

<1%

5

Submitted to North East Surrey College of Technology, Surrey

Student Paper

<1%

6

Submitted to Imperial College of Science, Technology and Medicine

Student Paper

<1%

7

do.sakura.ne.jp

Internet Source

<1%

8

codecrap.com

Internet Source

<1%

9	www.vr-online.ru Internet Source	<1 %
10	blog.harrix.org Internet Source	<1 %
11	swiki.cs.colorado.edu:3232 Internet Source	<1 %
12	www.xuelixue.cn Internet Source	<1 %
13	Submitted to Merrimack High School Student Paper	<1 %
14	snipt.net Internet Source	<1 %
15	Maillardet, . "Student projects", Introduction to Scientific Programming and Simulation Using R, 2009. Publication	<1 %
16	Submitted to University of Liverpool Student Paper	<1 %
17	Submitted to University of Sheffield Student Paper	<1 %
18	Submitted to Coventry University Student Paper	<1 %
19	googler700.blogspot.com Internet Source	<1 %

20

anas.pk
Internet Source

<1 %

21

www.slideshare.net
Internet Source

<1 %

22

blog.grkweb.com
Internet Source

<1 %

23

notesformsc.org
Internet Source

<1 %

24

js1k.com
Internet Source

<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off

Programming in C Assignment1

GRADEMARK REPORT

FINAL GRADE

78/100

GENERAL COMMENTS

Instructor

Exercise 1 44/50%

Flow chart shows logical planning of task and conforms to conventions. There was no requirement to prevent duplication of numbers, only to restrict the range.

Code was well presented and annotated, with good use of functions. It is normal practice to declare the prototype of the function before the main function and the definition after. Overall your program is extremely functional and robust and your code is logical and well implemented which includes checks on validity if input.

Exercise 2 24/ 30%

Flowchart is logical and demonstrates how program will function. A decision box is used where a branch in the code is required, the text inside this should be a question rather than a for/while construct.

As with exercise 1 code is very functional, and your user interface is very effective. However you have not considered as much validation on the inputs as you did with the previous exercise. Your use of functions is better in this exercise with the inclusion of prototypes.

Exercise 3 10/ 20%

You have made a good attempt at this problem, although there are some issues that still need ironing out. In development, consider simplification of the part of the task you are currently working on, get this simplified problem working, then expand on this to cover the actual problem, when dealing with a complex program it can be very hard to develop the entire feature you are working on straight away.

Checking for a win prior to changing player would solve the issue of the wrong player being told they had won.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38
