

Software Development Assignment 1

by Morgan Coombes

Submission date: 05-Apr-2019 11:49PM (UTC+0100)

Submission ID: 104253396

File name: 59982875.Morgan_Coombes_-_Software_Development_Assignment_1.docx (824.27K)

Word count: 5125

Character count: 26343

Software Development

ASSIGNMENT 1

MORGAN COOMBES - 144587

Contents

Task 1 – Higher Lower Game	2
Plan (Problem Solving)	2
Improvements to code	3
Testing	5
Errors while testing	5
Task 2 – Treasure Game	6
Plan (Problem Solving)	6
Improvements to code	Error! Bookmark not defined.
Testing	8
Errors while testing	9
Appendix.....	10
Screenshots	10
Higher Lower game screenshots.....	10
Treasure game screenshots	10
Code	19
Higher Lower – Code.....	24
Old Higher Lower Code	25
Treasure Game - Code	28

Task 1 – Higher Lower Game

Plan (Problem Solving)

Player 1 is to enter 7 numbers

This can be done by creating variables in the code to hold player 1's values, each value having a respective variable, I could then use 'cin' to allow player 1 to enter their value which will then be stored in the variable chosen.

The issue with this method is with efficiency, each value having to have it's own variable means I would have to declare 7 variables, each with different names to store player 1's values as each variable will only store 1 value, being the most recent one assigned to it; this makes things more complicated than they need to be.

Another way to store player 1's values would be to use an array, arrays are much more efficient and only require 1 name, they are initialised similar to regular variables, however they allow you to store multiple values which are decided when they are declared. I could make a single array which can hold all of player 1's values.

Using arrays allows this:

```
int value1;  
int value2;  
int value3; //etc
```

To be simplified to this:

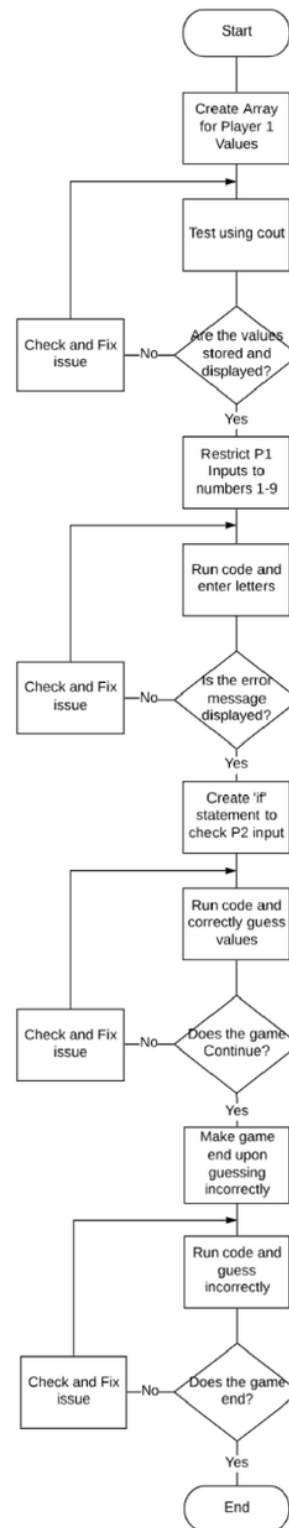
```
int gameValues[7]
```

Storing these values however, would work rather similarly, where 'cin' can be used to store the values in a declared variable, the array can't just be entered once as the program wouldn't know where the values would be stored in the array; so the code for storing values in an array would look something like this:

```
std::cin >> values[0]  
std::cin >> values[1] //etc
```

Once player 1 has entered their values, player 2 will now have to guess whether following values are higher or lower than the last, player 2 will be given the first value entered by player 1 as a starting point.

Player 2 should be able to guess with 'h' to guess higher, or 'l' to guess lower. I could implement an 'if' statement which will compare the last and next values to decide what happens, if player 2 guesses higher, then the game will continue if the value is higher, and if it is not higher, the game will end and player 2 will have lost, the opposite should happen if they guess lower, so if the next value is lower, then the game continues, and if it isn't, then the game will end and player 2 will lose.



Improvements to code

Player 1 inputs – When taking in player 1's inputs at first, I had 2 lines for each entry, one line with 'cout' to tell the user to input the value, then one line with 'cin' to allow the user to enter a value and have it stored in the array, my old code for storing player 1's values looked as such:

```
std::cout << "1st Value: ";

std::cin >> gameValues[0];

std::cout << "2nd Value: ";

std::cin >> gameValues[1];
```

This wasn't too bad but was more code than I needed as I would do the same for all 7 values, I ended up deleting this code and instead creating a loop that would run until certain conditions were met. The new code looked as follows:

```
9 for (int x = 0; x < 7; x++) {
    std::cout << "Please enter the next value: ";
    std::cin >> gameValues[x];
    while (gameValues[x] > 9 || gameValues[x] < 1) {
        std::cout << "Please enter a number between 1 and 9: ";
        std::cin >> gameValues[x];
    }
}
```

This reduced the amount of code by a fair bit whilst still accomplishing the same goal, this block would repeat until all 7 values were stored in the array. This loop also had a check for the values, to ensure that player 1 could not enter values outside of the range 1-9.

Player 2 guesses – When allowing player 2 to guess if values would be higher or lower than the last, I originally had quite a long if statement that would check if the value was higher or lower when player 2 entered 'h' or 'l', this method worked fine, however, this also could have been done in a lot less code. The old code made up the majority of the overall code in my program, as I would use the same block of code to compare each pair of values in the array (comparing values 1 and 2, then 2 and 3 etc), this code looked as follows:

```
while (gameValues[0] == gameValues[1]);
    std::cin >> guesses;
    if (guesses == 'h') {
        if (gameValues[0] < gameValues[1]) {
            std::cout << "Correct! The value is " << gameValues[1] << ", will the next
value be higher (h) or lower (l)?\n";
        }
        else if (gameValues[0] > gameValues[1]) {
            std::cout << "Incorrect! The value was " << gameValues[1] << ".\n";
        }
    }
    if (guesses == 'l') {
        if (gameValues[0] > gameValues[1]) {
            std::cout << "Correct! The value is " << gameValues[1] << ", will the next
value be higher (h) or lower (l)?\n";
```

```

    }
    else if (gameValues[0] < gameValues[1]) {
        std::cout << "Incorrect! The value was " << gameValues[1] << ".\n";
    }
}

```

This code was not very efficient as it would have been repeated 7 times (1 time for each value). I changed the code after to also run in a loop, which decreased the amount of code drastically, making my program much more efficient and much cleaner looking.

The new code looks like this:

```

for (x = 0; x < 7; x++) {
    std::cout << "\n";
    std::cin >> guesses;
    if (guesses == 'h') {
        if (gameValues[x] < gameValues[x + 1]) {
            std::cout << "Correct! The value is " << gameValues[x + 1] << ", will
the next value be higher (h) or lower (l)?\n";
        }
        else {
            std::cout << "Incorrect! The value was " << gameValues[x + 1] <<
".\n";
            x = 20;
        }
    }
    else if (guesses == 'l') {
        if (gameValues[x] > gameValues[x + 1]) {
            std::cout << "Correct! The value is " << gameValues[x + 1] << ",
will the next value be higher (h) or lower (l)?\n";
        }
        else {
            std::cout << "Incorrect! The value was " << gameValues[x + 1] <<
".\n";
            x = 20;
        }
    }
}
if (x == 21) {
    std::cout << "Sorry, you lose.\n";
}
else {
    std::cout << "Well done, you win!\n";
}
}

```

This code is much more efficient, as this block is only slightly larger than the old code, however this is a loop, so this code will loop for each value, so it will run until either all the values are guessed correctly, or a single value is guessed incorrectly.

Testing

Screenshots are included in the appendix

#	Test	Result	Solution (if required)
1	Enter values in the array and test 1 st value with cout	1 st value is displayed	No solution required
2	Enter values in the array and test 7 th value with cout	7 th value is displayed	No solution required
3	Enter letters when the game asks for numbers	Would not apply restriction, attempted restriction added extra input prompt (old code) Executable window fills with error text continuously (current code)	Did not solve problem
4	Enter numbers outside the range of 1-9	Error message would display, but it would still accept the value, essentially doing nothing	Changed code
5	Enter identical numbers consecutively	Game continues as normal until player 2's guesses	Did not solve problem
6	Guess one of player 1's values correctly	Game proceeds with congratulatory message	No solution required
7	Guess all of player 1's values correctly	Game continues past 7 values, and will set a large negative number as an 8 th value (old code) Could not get game to work properly (current code)	Did not solve problem
8	Guess one of player 1's values incorrectly	Game would not end, player 2 would still be shown the value and the game would continue as normal (old code) Game ends with game over message (current code)	Changed code

Errors while testing

Test 3 – When attempting to restrict player 1's inputs so that letters could not be entered, my code looked like this:

```
4 while (!(std::cin >> values))  
    {  
        std::cin.clear();  
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');  
        std::cout << "Please enter a number\n";  
    }
```

This code would not work in the issue, the restriction wouldn't actually apply, and letters were still accepted in the array, the line 'while(!(std::cin>>values))' caused the program to request another input from the user, which is not needed or used.

I did not fix the issue with this, and instead removed the block from my program as it was halting progress.

After I made changes to my code to make it more efficient, I decided to enter a letter instead of a number when the program was trying to collect player 1's values, this caused the window with the program running to fill up with the error message, the message would repeat continuously, essentially breaking my program.

Test 4 – When originally attempting to restrict player 1's inputs to the range 1-9, I could not get the code to work. The error message would pop up if numbers that were too high or too low were entered, however the value would still be accepted into the array. The code looked like this:

```
if (gameValues[0] < lowest || gameValues[0] > highest) {  
    std::cin.ignore();  
    std::cout << "Please enter a number from 1-9: ";  
}
```

I fixed this issue by making slight changes to my code, so now player 1 could not enter numbers outside the range 1-9 and would be prompted to enter a valid number. The message will now only display after an invalid entry and will not remain with following entries unless another invalid number is entered.

Test 7 – When I guesses all of the values correctly, the game didn't end and instead came up with an extra, 8th value which was '-858993460', both game over messages were also displayed simultaneously. I tried to resolve this issue by creating a variable called 'correct' which I set to 0, I would then have each correct guess add 1 onto this variable; I then set the condition for the congratulatory game over message to:

```
else if (correct == 7) {  
    std::cout << "Well done, you win!\n";  
}
```

Even with this change, the game still wouldn't end and there would be an extra value, so I then tried to change the condition to end if correct was not equal to 7 as such:

```
if (correct != 7) {  
    std::cout << "Sorry, you lose.\n";  
}
```

Task 2 – Treasure Game

Plan (Problem Solving)

Player 1 will enter X and Y co-ordinates which will be used to place a treasure in a grid. I can create an array to store the co-ordinates, having a different array for X and Y, player 1 will enter 3 sets of co-ordinates so the arrays will only need to hold 3 values each.

These co-ordinates will place a treasure on an 8x8 grid, this grid can be made using a dimensional array, this would look something like this: 'char grid[9][9];' This will declare a grid with a size of 9x9, the grid only needs to be 8x8, however I will use the extra row and column to label the grid, guiding the players.

This grid can be displayed using a loop, which can print rows and columns which will have to be declared as variables, I declared them within the loop as I wouldn't need to use them outside of the loop. The rows will be printed until a condition is met, the same can be applied to the columns; so, if I want to print my full grid out (9x9), I would make rows print until the value for the row integer reaches 9, the same can be done for the columns. The statement would look something like this:

```

2
for (int row = 0; row < 9; row++) {
    for (int column = 0; column < 9; column++) {
        std::cout << grid[row][column] << " ";
    }
    cout << "\n";
}

```

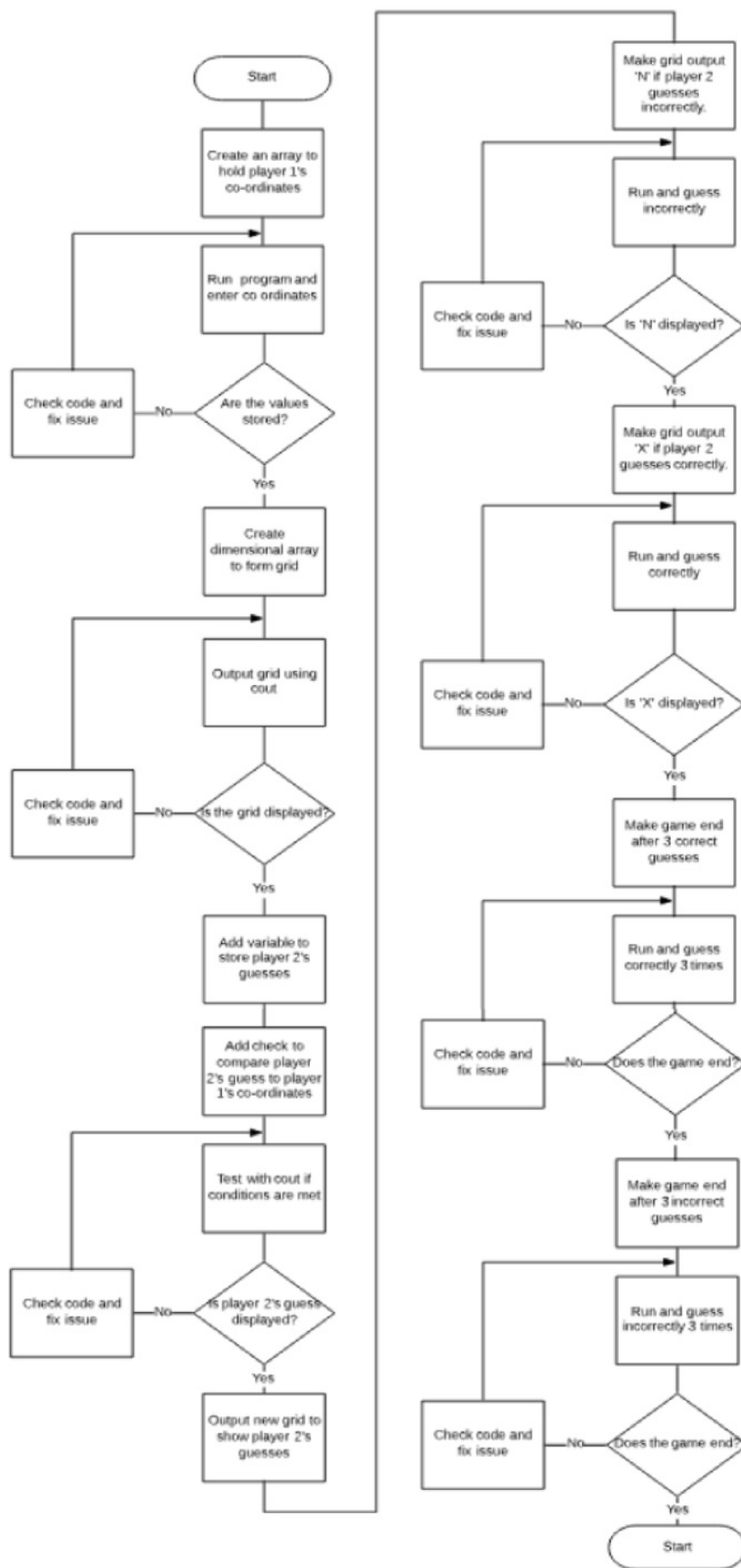
That block would set the conditions for the row and column, they would then print out until they reach 9, as each row that is outputted will add 1 onto the row and the same goes for the columns. The block also adds spacing between the rows and columns.

Player 2 will have to guess where player 1 placed the treasure, I can create variables to store their guess for each axis, however this doesn't need to be an array despite player 2 having to guess 3 co-ordinates. Arrays would have to be used if I needed to keep player 2's guess after they find the treasure, however this is not the case, I can instead use a regular variable which can only store 1 value, only the last value will be kept, so if player 2 is prompted to guess the X co-ordinates, and Y co-ordinates, these values will then be placed in the variable, this is where I would check the variable values against the first values in the X array and Y array for player 1's co-ordinates, if player 2 guesses incorrectly, then they can try again, the values they enter this time will then replace the other ones in the variable, since they were incorrect, they aren't needed anyway. If player 2 guesses the co-ordinates correctly, then they will be prompted to guess where the second treasure is, and again, the values will overwrite the old ones in the guess variables, since the previous treasure has already been found, the values are no longer needed, the same would be repeated for the 3rd treasure. This can all be done in a loop, so it will automatically repeat until certain conditions are met.

Player 2 will get 3 lives, allowing them to guess incorrectly 3 times before the game ends, each incorrect guess will decrease their lives count, to make this work I would make a variable to hold player 2's lives and set the value to 3, then for each incorrect guess player 2 makes, I can decrease the value of this variable easily, as such: 'lives --' That will decrease the count by 1, only on incorrect guesses. The game could then end on an 'if' statement that will end the game if the value of the 'lives' integer reaches 0.

The goal of player 2 is to guess all sets of co-ordinates correctly and win the game, because of this the game should include a way of tracking how many treasures player 2 has found, this could again be done with the help of a variable, though this one would start at 0, then each treasure that is found will add to the value of this integer, this can also be done easily as such: 'found++' This will add 1 onto the value of the variable for each treasure found, the game can be ended using an 'if' statement when the value of 'found' reaches 3.

When player 2 inputs their guess, either an 'X' or 'N' should be displayed on the grid, depending on if they find the treasure or not, an 'X' will be displayed if the co-ordinates match where player 1 placed the treasure, and 'N' will be placed if the guess was incorrect. A new grid should be displayed for each guess that is made, the new grid will then include the 'X' or 'N'.



Testing

#	Test	Result	Solution (if required)
1	Enter values in X axis array, test 1 st value with cout	1 st X value is displayed	No solution required
2	Enter values in X axis array, test 3 rd value with cout	3 rd X value is displayed	No solution required
3	Enter values in Y axis array, test 1 st value with cout	1 st Y value is displayed	No solution required
4	Enter values in Y axis array, test 3 rd value with cout	3 rd Y value is displayed	No solution required
5	Run code and see if grid outputs correctly	Full grid is displayed	No solution required
6	Enter identical co-ordinates for two treasures	Treasure is still placed, when the co-ordinates are guessed, the game doesn't move past the 2 nd treasure	Did not resolve issue
7	Guess co-ordinates incorrectly once	An 'N' is placed at the incorrect co-ordinates and a life is deducted	No solution required
8	Guess co-ordinates incorrectly 3 times	3 'N's are placed at the incorrect co-ordinates and the game ends	No solution required
9	Guess co-ordinates correctly	An 'X' is revealed at the correct co-ordinates and 1 is added to the 'found' variable	No solution required
10	Guess co-ordinates correctly 3 times	3 'X's are revealed at the correct co-ordinates and the game ends with congratulatory message	No solution required
11	Grid updating with 'X' upon guessing correctly	A new grid is displayed with the 'X' revealed where the treasure is	No solution required
12	Grid updating with 'N' upon guessing incorrectly	A new grid is displayed with the 'N' placed at the incorrect co-ordinates	No solution required
13	Enter co-ordinates larger than the grid boundaries	Treasure is still placed, though the 'X' isn't revealed when a correct guess is made	Did not resolve issue

Errors while testing

When I entered identical co-ordinates for two treasures, the treasures were still placed, though when it came to guessing the co-ordinates, the first treasure was found, then the 2nd, but the game would not move past the 2nd treasure, it still counted as a correct guess but went back to that treasure again, while adding 1 to the 'found' variable, so the game was essentially won by finding 2 treasures.

When I entered co-ordinates that were larger values than the grid would hold, the treasure was still placed, and it could be guessed correctly or incorrectly by player 2, the 'X' or 'N' would not be visible as it would try and place them at the co-ordinates entered despite them not being visible. I also set all the values to be off the grid's boundaries, and after guessing all the values correctly, the program crashed.

[Appendix](#)

[Screenshots](#)

[Higher Lower game screenshots](#)

Improvements

Screenshot of old array code

```
std::cout << "Player 1, please enter 7, single digit values: \n"; //promts first player to enter their values
string gameValues[7]; //created array for holding player 1's values
int values;
std::cout << "1st Value: ";
std::cin >> gameValues[0]; //stores 1st value entered by player 1
std::cout << "2nd Value: ";
std::cin >> gameValues[1];
std::cout << "3rd Value: ";
std::cin >> gameValues[2];
std::cout << "4th Value: ";
std::cin >> gameValues[3];
std::cout << "5th Value: ";
std::cin >> gameValues[4];
std::cout << "6th Value: ";
std::cin >> gameValues[5];
std::cout << "7th Value: ";
std::cin >> gameValues[6];
```

Screenshot of new array code

```
#include "pch.h"
#include <iostream>
#include <string>
using namespace std;

int main()
{
    std::cout << "Player 1, please enter 7 values from 1-9.\n\n";
    int gameValues[7]; // This names my array and declares how many values will be stored
    const int lowest = 1; // This is to limit player 1's values so they can be no lower than 1
    const int highest = 9; // This is to limit player 1's values so they can be no higher than 9

    for (int x = 0; x < 7; x++) { //This causes my array to loop until the 7 values are entered
        std::cout << "Please enter the next value: ";
        std::cin >> gameValues[x]; //This places the value into the array
    }
}
```

Screenshot(s) for [Test 1](#)

```
Player 1, please enter 7 values from 1-9.
```

```
Please enter the next value: 3
Please enter the next value: 5
Please enter the next value: 2
Please enter the next value: 7
Please enter the next value: 5
Please enter the next value: 8
Please enter the next value: 2
First value: 3
```

Screenshot(s) for [Test 2](#)

```
Player 1, please enter 7 values from 1-9.  
  
Please enter the next value: 2  
Please enter the next value: 4  
Please enter the next value: 7  
Please enter the next value: 5  
Please enter the next value: 7  
Please enter the next value: 3  
Please enter the next value: 9  
Last value: 9
```

Screenshot(s) for [Test 3](#)

Entering letters before restriction was put in place

```
Player 1, please enter 7, single digit values:  
1st Value: 10  
2nd Value: h  
3rd Value: hello  
4th Value: 0  
5th Value: 2  
6th Value: 3  
7th Value: 1  
1
```

Entering a letter after changing array method and restricting numbers

[illegible]

Screenshot(s) for [Test 4](#)

First attempt at restricting values to 1-9

```
Player 1, please enter 7, single digit values:  
1st Value: 10  
Please enter a number from 1-9: 2nd Value: 0  
Please enter a number from 1-9: 3rd Value: 3  
4th Value: 4  
5th Value: 5  
6th Value: 6  
7th Value: 11
```

Other attempts at restricting size of numbers entered

```
Player 1, please enter 7 values from 1-9.  
  
Please enter the next value: 10  
Please enter a number from 1-9: Please enter the next value: Please enter a number from 1-9: _
```

```
Player 1, please enter 7 values from 1-9.  
  
Please enter the next value: 10  
Please enter a number from 1-9: Please enter the next value: Please enter a number from 1-9: Please enter the next value:  
: Please enter a number from 1-9: Please enter the next value: _
```

Working restriction on size of numbers that can be entered

```
Player 1, please enter 7 values from 1-9.  
  
Please enter the next value: 0  
Please enter a number between 1 and 9: 1  
Please enter the next value: 10  
Please enter a number between 1 and 9: 5  
Please enter the next value: 7  
Please enter the next value: 2  
Please enter the next value: 3  
Please enter the next value: 1  
Please enter the next value: 5  
Player 2, the 1st value is 1, will the following value be higher (h) or lower (l)?
```

Screenshot(s) for [Test 5](#)

```
Player 1, please enter 7 values from 1-9.

Please enter the next value: 3
Please enter the next value: 3
Please enter the next value: 5
Please enter the next value: 6
Please enter the next value: 7
Please enter the next value: 8
Please enter the next value: 9
Player 2, the 1st value is 3, will the following value be higher (h) or lower (l)?
h
Incorrect! The value was 3.
Sorry, you lose.
```

```
Player 1, please enter 7 values from 1-9.

Please enter the next value: 3
Please enter the next value: 3
Please enter the next value: 4
Please enter the next value: 5
Please enter the next value: 6
Please enter the next value: 7
Please enter the next value: 8
Player 2, the 1st value is 3, will the following value be higher (h) or lower (l)?
l
Incorrect! The value was 3.
Sorry, you lose.
```

Screenshot(s) for [Test 6](#)

```
Player 1, please enter 7 values from 1-9.

Please enter the next value: 2
Please enter the next value: 4
Please enter the next value: 6
Please enter the next value: 8
Please enter the next value: 6
Please enter the next value: 4
Please enter the next value: 2
Player 2, the 1st value is 2, will the following value be higher (h) or lower (l)?
h
Correct! The value is 4, will the next value be higher (h) or lower (l)?
l
```

Screenshot(s) for [Test 7](#)

```
Player 1, please enter 7 values from 1-9.

Please enter the next value: 2
Please enter the next value: 4
Please enter the next value: 6
Please enter the next value: 8
Please enter the next value: 6
Please enter the next value: 4
Please enter the next value: 2
Player 2, the 1st value is 2, will the following value be higher (h) or lower (l)?
h
Correct! The value is 4, will the next value be higher (h) or lower (l)?
h
Correct! The value is 6, will the next value be higher (h) or lower (l)?
h
Correct! The value is 8, will the next value be higher (h) or lower (l)?
l
Correct! The value is 6, will the next value be higher (h) or lower (l)?
l
Correct! The value is 4, will the next value be higher (h) or lower (l)?
l
Correct! The value is 2, will the next value be higher (h) or lower (l)?
l
Correct! The value is -858993460, will the next value be higher (h) or lower (l)?
Well done, you win!
```

Screenshot(s) for [Test 8](#)

```
Player 1, please enter 7 values from 1-9.

Please enter the next value: 2
Please enter the next value: 4
Please enter the next value: 6
Please enter the next value: 8
Please enter the next value: 6
Please enter the next value: 4
Please enter the next value: 2
Player 2, the 1st value is 2, will the following value be higher (h) or lower (l)?
h
Correct! The value is 4, will the next value be higher (h) or lower (l)?
l
Incorrect! The value was 6.
Sorry, you lose.
```

[Treasure game screenshots](#)

Screenshot(s) for [Test 1](#)

```
Player 1, place 3 treasures now:  
  
X: 2  
Y: 3  
1st treasure placed.  
  
X: 4  
Y: 5  
2nd treasure placed.  
  
X: 6  
Y: 7  
3rd treasure placed.  
  
First X co-ordinate: 2
```

Screenshot(s) for [Test 2](#)

```
Player 1, place 3 treasures now:  
  
X: 1  
Y: 2  
1st treasure placed.  
  
X: 3  
Y: 4  
2nd treasure placed.  
  
X: 5  
Y: 6  
3rd treasure placed.  
  
Last X co-ordinate: 5
```

Screenshot(s) for [Test 3](#)

Player 1, place 3 treasures now:

X: 2
Y: 3
1st treasure placed.

X: 4
Y: 5
2nd treasure placed.

X: 6
Y: 7
3rd treasure placed.

First Y co-ordinate: 3

Screenshot(s) for [Test 4](#)

Player 1, place 3 treasures now:

X: 2
Y: 3
1st treasure placed.

X: 4
Y: 5
2nd treasure placed.

X: 6
Y: 7
3rd treasure placed.

Last Y co-ordinate: 7

Screenshot(s) for [Test 5](#)

```
Player 1, place 3 treasures now:

X: 2
Y: 2
1st treasure placed.

X: 3
Y: 3
2nd treasure placed.

X: 5
Y: 5
3rd treasure placed.

.  0  1  2  3  4  5  6  7
0  -  -  -  -  -  -  -  -
1  -  -  -  -  -  -  -  -
2  -  -  -  -  -  -  -  -
3  -  -  -  -  -  -  -  -
4  -  -  -  -  -  -  -  -
5  -  -  -  -  -  -  -  -
6  -  -  -  -  -  -  -  -
7  -  -  -  -  -  -  -  -
Player 2, where is the treasure?
X:
```

Screenshot(s) for [Test 6](#)

```
Player 1, place 3 treasures now:

X: 2
Y: 3
1st treasure placed.

X: 2
Y: 3
2nd treasure placed.

X: 7
Y: 7
3rd treasure placed.

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 2
Y: 3
Correct, you've found the first treasure!
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - X - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 2
Y: 3
Correct, you've found the first treasure!
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - X - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 2
Y: 3
Correct, you've found the first treasure!
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - X - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Congratulations, you've found all the treasure, you win.
```

Screenshot(s) for [Test 7](#)

```
Player 1, place 3 treasures now:

X: 2
Y: 3
1st treasure placed.

X: 4
Y: 5
2nd treasure placed.

X: 6
Y: 7
3rd treasure placed.

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -

Player 2, where is the treasure?
X: 2
Y: 4
Incorrect, try again.
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - N - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -

Player 2, where is the treasure?
X: _
```

Screenshot(s) for [Test 8](#)

```
Player 1, place 3 treasures now:

X: 2
Y: 3
1st treasure placed.

X: 4
Y: 5
2nd treasure placed.

X: 6
Y: 7
3rd treasure placed.

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -

Player 2, where is the treasure?
X: 2
Y: 4
Incorrect, try again.
```

```

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - N - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 5
Y: 4
Incorrect, try again.
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - N - - -
2 - - - - - - -
3 - - - - - - -
4 - - - N - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 6
Y: 3
Incorrect, try again.
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - N - - -
2 - - - - - - -
3 - - - - - - -
4 - - - N - - -
5 - - N - - - -
6 - - - - - - -
7 - - - - - - -
You lose, better luck next time.

```

Screenshot(s) for [Test 9](#)

```

Player 1, place 3 treasures now:

X: 2
Y: 2
1st treasure placed.

X: 4
Y: 4
2nd treasure placed.

X: 6
Y: 6
3rd treasure placed.

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 2
Y: 2
Correct, you've found the first treasure!
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - X - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: -

```

Screenshot(s) for [Test 10](#)

```
Player 1, place 3 treasures now:
X: 2
Y: 2
1st treasure placed.
X: 4
Y: 4
2nd treasure placed.
X: 6
Y: 6
3rd treasure placed.

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 2
Y: 2
Correct, you've found the first treasure!
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - X - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 4
Y: 4
Correct, you've found the second treasure!
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - X - - - - -
2 - - - - - - -
3 - - - X - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -
Player 2, where is the treasure?
X: 6
Y: 6
Correct, you've found the third treasure!
. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - X - - - - -
2 - - - - - - -
3 - - - X - - -
4 - - - - - - -
5 - - - - X - -
6 - - - - - - -
7 - - - - - - -
Congratulations, you've found all the treasure, you win.
```

Screenshot(s) for [Test 11](#)

```
Player 1, place 3 treasures now:

X: 2
Y: 2
1st treasure placed.

X: 4
Y: 4
2nd treasure placed.

X: 6
Y: 6
3rd treasure placed.

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -

Player 2, where is the treasure?
X: 2
Y: 2
Correct, you've found the first treasure!

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - X - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -

Player 2, where is the treasure?
X: _
```

Screenshot(s) for [Test 12](#)

```
Player 1, place 3 treasures now:

X: 2
Y: 3
1st treasure placed.

X: 4
Y: 5
2nd treasure placed.

X: 6
Y: 7
3rd treasure placed.

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -

Player 2, where is the treasure?
X: 2
Y: 4
Incorrect, try again.

. 0 1 2 3 4 5 6 7
0 - - - - - - -
1 - - - N - - - -
2 - - - - - - -
3 - - - - - - -
4 - - - - - - -
5 - - - - - - -
6 - - - - - - -
7 - - - - - - -

Player 2, where is the treasure?
X: _
```

Screenshot(s) for [Test 13](#)

```
Player 1, place 3 treasures now:

X: 10
Y: 10
1st treasure placed.

X: 7
Y: 6
2nd treasure placed.

X: 2
Y: 4
3rd treasure placed.

.  0  1  2  3  4  5  6  7
0  -  -  -  -  -  -  -  -
1  -  -  -  -  -  -  -  -
2  -  -  -  -  -  -  -  -
3  -  -  -  -  -  -  -  -
4  -  -  -  -  -  -  -  -
5  -  -  -  -  -  -  -  -
6  -  -  -  -  -  -  -  -
7  -  -  -  -  -  -  -  -
Player 2, where is the treasure?
X: 10
Y: 10
Correct, you've found the first treasure!
.  0  1  2  3  4  5  6  7
0  -  -  -  -  -  -  -  -
1  -  -  -  -  -  -  -  -
2  -  -  -  -  -  -  -  -
3  -  -  -  -  -  -  -  -
4  -  -  -  -  -  -  -  -
5  -  -  -  -  -  -  -  -
6  -  -  -  -  -  -  -  -
7  -  -  -  -  -  -  -  -
Player 2, where is the treasure?
X:
```

[Code](#)

Higher Lower – Code

```
#include "pch.h"
#include <iostream>
using namespace std;

int main()
{
    std::cout << "Player 1, please enter 7 values from 1-9.\n\n\n";
    int gameValues[7]; // This names my array and declares how many values will be
    stored
    const int lowest = 1; // This is to limit player 1's values so they can be no
    lower than 1
    const int highest = 9; // This is to limit player 1's values so they can be no
    higher than 9

    for (int x = 0; x < 7; x++) { //This causes my array to loop until the 7 values
    are entered
        std::cout << "Please enter the next value: ";
        std::cin >> gameValues[x]; //This places the value into the array after
    the values are checked
        while (gameValues[x] > 9 || gameValues[x] < 1) {
            std::cout << "Please enter a number between 1 and 9: ";
            std::cin >> gameValues[x];
        }
    }
    char guesses;
    int correct = 0;
    int x;
    std::cout << "Player 2, the 1st value is " << gameValues[0] << ", will the
    following value be higher (h) or lower (l)?\n";
    for (x = 0; x < 7; x++) {

        std::cout << "\n";
        std::cin >> guesses;
        if (guesses == 'h') { //This check is run if player 2 enters 'h' to
    guess higher
            if (gameValues[x] < gameValues[x + 1]) {
                std::cout << "Correct! The value is " << gameValues[x + 1]
    << ", will the next value be higher (h) or lower (l)?\n"; //If the value is higher,
    this message is displayed
                correct++;
            }
            else {
                std::cout << "Incorrect! The value was " << gameValues[x +
    1] << ".\n"; //If the value is lower, this message is displayed
                //guesses = 'X';
                x = 20;
            }
        }
        else if (guesses == 'l') { //This check is run if player 2 enters 'h' to
    guess higher
            if (gameValues[x] > gameValues[x + 1]) {
                std::cout << "Correct! The value is " << gameValues[x + 1]
    << ", will the next value be higher (h) or lower (l)?\n"; //If the value is higher,
    this message is displayed
                correct++;
            }
            else {
                std::cout << "Incorrect! The value was " << gameValues[x +
    1] << ".\n"; //If the value is lower, this message is displayed
            }
        }
    }
}
```

```

        x = 20;
    }
}
}
if (correct != 7) {
    std::cout << "Sorry, you lose.\n"; //This message displays if player
2guesses incorrectly
}
else if (correct == 7) {
    std::cout << "Well done, you win!\n"; //This message displays if player 2
guesses correctly for all values
}
}

```

Old Higher Lower Code

```

#include "pch.h"
#include <string>
#include <iostream>
using namespace std;

int main()
{
    int lowest = 1;
    int highest = 9;
    std::cout << "Player 1, please enter 7, single digit values: \n"; //promts
first player to enter their values
    string gameValues[7]; //created array for holding player 1's values
    int values;
    std::cout << "1st Value: ";
    std::cin >> gameValues[0]; //stores 1st value entered by player 1
    std::cout << "2nd Value: ";
    std::cin >> gameValues[1];
    std::cout << "3rd Value: ";
    std::cin >> gameValues[2];
    std::cout << "4th Value: ";
    std::cin >> gameValues[3];
    std::cout << "5th Value: ";
    std::cin >> gameValues[4];
    std::cout << "6th Value: ";
    std::cin >> gameValues[5];
    std::cout << "7th Value: ";
    std::cin >> gameValues[6];
    3
    while (!(std::cin >> values))
    {
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
        std::cout << "Please enter a number\n";
    }
    char guesses;
    std::cout << "\n";
    std::cout << "Player 2, the 1st value is " << gameValues[0] << ", will the
following value be higher (h) or lower (l)?\n";
    while (gameValues[0] == gameValues[1]);
    std::cin >> guesses;
    if (guesses == 'h') {
        if (gameValues[0] < gameValues[1]) {
            std::cout << "Correct! The value is " << gameValues[1] << ", will
the next value be higher (h) or lower (l)?\n";

```

```

    }
    else if (gameValues[0] > gameValues[1]) {
        std::cout << "Incorrect! The value was " << gameValues[1] <<
".\n";
    }
}
if (guesses == 'l') {
    if (gameValues[0] > gameValues[1]) {
        std::cout << "Correct! The value is " << gameValues[1] << ", will
the next value be higher (h) or lower (l)?\n";
    }
    else if (gameValues[0] < gameValues[1]) {
        std::cout << "Incorrect! The value was " << gameValues[1] <<
".\n";
    }
}
while (gameValues[1] == gameValues[2]);
std::cin >> guesses;
if (guesses == 'h') {
    if (gameValues[1] < gameValues[2]) {
        std::cout << "Correct! The value is " << gameValues[2] << ", will
the next value be higher (h) or lower (l)?\n";
    }
    else if (gameValues[1] > gameValues[2]) {
        std::cout << "Incorrect! The value was " << gameValues[2] <<
".\n";
    }
}
if (guesses == 'l') {
    if (gameValues[1] > gameValues[2]) {
        std::cout << "Correct! The value is " << gameValues[2] << ", will
the next value be higher (h) or lower (l)?\n";
    }
    else if (gameValues[1] < gameValues[2]) {
        std::cout << "Incorrect! The value was " << gameValues[2] <<
".\n";
    }
}
while (gameValues[0] == gameValues[1]);
std::cin >> guesses;
if (guesses == 'h') {
    if (gameValues[2] < gameValues[3]) {
        std::cout << "Correct! The value is " << gameValues[3] << ", will
the next value be higher (h) or lower (l)?\n";
    }
    else if (gameValues[2] > gameValues[3]) {
        std::cout << "Incorrect! The value was " << gameValues[3] <<
".\n";
    }
}
if (guesses == 'l') {
    if (gameValues[2] > gameValues[3]) {
        std::cout << "Correct! The value is " << gameValues[3] << ", will
the next value be higher (h) or lower (l)?\n";
    }
    else if (gameValues[2] < gameValues[3]) {
        std::cout << "Incorrect! The value was " << gameValues[3] <<
".\n";
    }
}
while (gameValues[3] == gameValues[4]);
std::cin >> guesses;

```

```

        if (guesses == 'h') {
            if (gameValues[3] < gameValues[4]) {
                std::cout << "Correct! The value is " << gameValues[4] << ", will
the next value be higher (h) or lower (l)?\n";
            }
            else if (gameValues[3] > gameValues[4]) {
                std::cout << "Incorrect! The value was " << gameValues[4] <<
".\n";
            }
        }
        if (guesses == 'l') {
            if (gameValues[3] > gameValues[4]) {
                std::cout << "Correct! The value is " << gameValues[4] << ", will
the next value be higher (h) or lower (l)?\n";
            }
            else if (gameValues[3] < gameValues[4]) {
                std::cout << "Incorrect! The value was " << gameValues[4] <<
".\n";
            }
        }
        while (gameValues[4] == gameValues[5]);
        std::cin >> guesses;
        if (guesses == 'h') {
            if (gameValues[4] < gameValues[5]) {
                std::cout << "Correct! The value is " << gameValues[5] << ", will
the next value be higher (h) or lower (l)?\n";
            }
            else if (gameValues[4] > gameValues[5]) {
                std::cout << "Incorrect! The value was " << gameValues[5] <<
".\n";
            }
        }
        if (guesses == 'l') {
            if (gameValues[4] > gameValues[5]) {
                std::cout << "Correct! The value is " << gameValues[5] << ", will
the next value be higher (h) or lower (l)?\n";
            }
            else if (gameValues[4] < gameValues[5]) {
                std::cout << "Incorrect! The value was " << gameValues[5] <<
".\n";
            }
        }
        while (gameValues[5] == gameValues[6]);
        std::cin >> guesses;
        if (guesses == 'h') {
            if (gameValues[5] < gameValues[6]) {
                std::cout << "Correct! The value is " << gameValues[6] << ", will
the next value be higher (h) or lower (l)?\n";
            }
            else if (gameValues[5] > gameValues[6]) {
                std::cout << "Incorrect! The value was " << gameValues[6] <<
".\n";
            }
        }
        if (guesses == 'l') {
            if (gameValues[5] > gameValues[6]) {
                std::cout << "Correct! The value is " << gameValues[6] << ", You
Win!\n";
            }
            else if (gameValues[5] < gameValues[6]) {
                std::cout << "Incorrect! The value was " << gameValues[6] <<
".\n";
            }
        }

```

```

    }
}
}

```

Treasure Game - Code

```

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

int main()
{
    int axisX[3]; //Declares an array for storing player 1's X axis values
    int axisY[3]; //Declares an array for storing player 1's Y axis values
    int lives = 3; //Declares an integer for player 2's lives (3), this decreases
    with incorrect guesses
    int found = 0; //Declares an integer for treasures found by player 2, increasing
    for each treasure found
    std::cout << "Player 1, place 3 treasures now:\n\n"; //Prompts player to
    place treasure at chosen co-ordinates
    std::cout << "X: ";
    std::cin >> axisX[0]; //Stores users input in X axis array
    std::cout << "Y: ";
    std::cin >> axisY[0]; //Stores users input in Y axis array
    std::cout << "1st treasure placed.\n\n"; //This message displays once the co-
    ordinates are entered
    //The same is repeated for treasures 2 and 3
    std::cout << "X: ";
    std::cin >> axisX[1];
    std::cout << "Y: ";
    std::cin >> axisY[1];
    std::cout << "2nd treasure placed.\n\n";
    std::cout << "X: ";
    std::cin >> axisX[2];
    std::cout << "Y: ";
    std::cin >> axisY[2];
    std::cout << "3rd treasure placed.\n\n";
    char grid[9][9] = { { '.', '0', '1', '2', '3', '4', '5', '6', '7' }, //This declares my
    8x8 grid array
    { '0', '-', '-', '-', '-', '-', '-', '-', '-', '-' },
    { '1', '-', '-', '-', '-', '-', '-', '-', '-', '-' },
    { '2', '-', '-', '-', '-', '-', '-', '-', '-', '-' },
    { '3', '-', '-', '-', '-', '-', '-', '-', '-', '-' },
    { '4', '-', '-', '-', '-', '-', '-', '-', '-', '-' },
    { '5', '-', '-', '-', '-', '-', '-', '-', '-', '-' },
    { '6', '-', '-', '-', '-', '-', '-', '-', '-', '-' },
    { '7', '-', '-', '-', '-', '-', '-', '-', '-', '-' } };
    for (int row = 0; row < 9; row++) { //Will add a row until the variable value
    reaches 9 (the limit of the array)
        for (int column = 0; column < 9; column++) { //Will add a column until
        the variable value reaches 9 (the limit of the array)
            std::cout << grid[row][column] << " "; //Outputs the rows and
            columns
        }
        cout << "\n";
    }
    while (lives > 0 && found < 3) { //The game will continue until player 2 runs
    out of lives, or finds all 3 treasures

```

```

        std::cout << "Player 2, where is the treasure?\n";
        int guessX; //Stores player 2's most recent X axis guess
        int guessY; //Stores player 2's most recent Y axis guess
        std::cout << "X: ";
        std::cin >> guessX; //Player 2 enters their guess for the X axis
        std::cout << "Y: ";
        std::cin >> guessY; //Player 2 enters their guess for the Y axis
        if (axisX[0] == guessX && axisY[0] == guessY) { //Checks if player 2's
guess is equal to where player 1 placed the first treasure
            std::cout << "Correct, you've found the first treasure!" << endl;
//This message is displayed if the condition is met (if player 2 finds the treasure)
            grid[guessX][guessY] = 'X'; //Places an 'X' if player 2's guess
matches player 1's co-ordinates
            found++; //Increases the count variable to keep track of how many
treasures are found
        }
        else if (axisX[1] == guessX && axisY[1] == guessY) { //Checks if player
2's guess is equal to where player 1 placed the second treasure
            std::cout << "Correct, you've found the second treasure!" <<
endl; //This message shows if the condition is met
            grid[guessX][guessY] = 'X'; //Places an 'X' if the co-ordinates
match
            found++; //Adds 1 onto the 'found' variable
        }
        else if (axisX[2] == guessX && axisY[2] == guessY) {
            std::cout << "Correct, you've found the third treasure!" << endl;
            grid[guessX][guessY] = 'X';
            found++;
        }
        else { //If none of the other conditions are met, then this block is run
            std::cout << "Incorrect, try again." << endl; //This message is
displayed if player 2's guess doesn't match player 1's co-ordinates
            grid[guessX][guessY] = 'N'; //Places an 'N' at the incorrect co-
ordinates player 2 entered
            lives--; //Decreases the count of the lives variable for each
incorrect guess
        }

        for (int row = 0; row < 9; row++) { //This block will output a new grid
for each of player 2's guesses, updating the grid with an 'X' or 'N', depending on
their guess
            for (int column = 0; column < 9; column++) {
                std::cout << grid[row][column] << " ";
            }
            cout << "\n";
        }
    }
    if (lives == 0) { //Game will end once player 2 guesses incorrectly 3 times
        std::cout << "You lose, better luck next time.\n"; //This message is
displayed when player 2 loses
    }
    else { //Game will end once player 2 finds all the treasures
        std::cout << "Congratulations, you've found all the treasure, you
win.\n"; //This message is displayed when player 2 wins
    }
}
}

```

Software Development Assignment 1

ORIGINALITY REPORT

3%

SIMILARITY INDEX

3%

INTERNET SOURCES

0%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to University of Lincoln

Student Paper

1%

2

Submitted to Imperial College of Science,
Technology and Medicine

Student Paper

1%

3

Submitted to Columbia Basin College

Student Paper

<1%

4

www.daniweb.com

Internet Source

<1%

5

www.learncpp.com

Internet Source

<1%

6

caveofprogramming.com

Internet Source

<1%

7

stackoverflow.com

Internet Source

<1%

8

www.cppfrance.com

Internet Source

<1%

9

codegur.com

--	--	--	--

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Software Development Assignment 1

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

46/100

Instructor

Exercise 1 25/ 50%

Flow chart shows logical planning of how you intend to tackle the task and conforms to conventions.

Code was well presented and annotated, however you created constants for the highest and lowest values, but then did not use them. Note that the brief said to accept values in the range 1-10, where your code is taking values between 2 and 8.

The reason your program does not report a win correctly is because correct will be 6, not 7, as player 2 is given the first number and then guesses about the next 6 values. The section where player 2 makes the guesses would have been more efficient if implemented using a loop.

Use of functions would have improved readability.

In testing you should have included boundary tests for the player 1 values to ensure you would take values between 1 and 10 and reject all others, tests at 0, 1, 10 and 11 would have allowed you to see the issues you had with the program.

Exercise 2 21/ 30%

Again your flowchart demonstrates how you intended to tackle the problem and you have also discussed how you developed it.

You could have used a two dimensional array for the coordinates of the treasures, but better still store them in the playing grid itself, just write code that does not print them out. Consider validation of these numbers to ensure they sit within the treasure map range.

The numbers around the grid do not reflect the actual coordinates player two has entered, note the entered coordinates (4,4) have resulted in an X in position (3,3) in your grid, this would be easily resolved by simply changing the numbers around the grid.

Again as for previous exercise consider the use of functions.

Exercise 3 0/ 20%

You have not attempted this exercise.

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
