

Software Development

by Samia Assebab

FILE	5773_SAMIA_ASSEBAB_SOFTWARE_DEVELOPMENT_178138_66123954 6.DOCX (373.28K)		
TIME SUBMITTED	12-APR-2019 08:53PM (UTC+0100)	WORD COUNT	3302
SUBMISSION ID	104703677	CHARACTER COUNT	20741

SOFTWARE DEVELOPMENT

ACCA4017

ABSTRACT

Code compilation of three games utilising visual studio, assessing the problems faced upon attempt and the problem-solving methods to implement logical, readable code.

Samia Assebab

P151305

Contents

Introduction.....	2
“High-Low Guessing” Game	3
Planning	3
Trial and Error	4
Problem Solving	5
“Hidden Treasure” Game	6
Planning	6
Trial and Error	6
Problem Solving	7
Findings	8
Conclusion.....	9
Appendix.....	9
Attempt 1: “High-Low Game”	9
Attempt 2: “High-Low Game”	12
Final Attempt: “High-Low Game”	14
Attempt 1: “Treasure Hunt Game ”	17
Attempt 2: “Treasure Hunt Game ”	20
Final : “Treasure Hunt Game ”	23
Figure 1- Specifications	3
Figure 2-Implementation	3
Figure 3 -Player 1 String Array.....	4
Figure 6 – Problem Solving Notes	5
Figure 7- Final Implimentation.....	5
Figure 8- Grid String Array	6
Figure 9- 1D Array	7
Figure 10- 2D Array	7
Figure 12-Final Implementation	7
Figure 13- Integer Array	8

Introduction

This report will explore the authors attempt to complete three games using computing language C++, the report will explain planning methodologies before writing the code, the processes involved in properly articulating the desired outcome. The hardships faced by the author and effective strategies for remedy.

The first exercise involves creating a game comprising a two-user function, whereby player 1 enters 7 integer values which are systematically displayed after player two is prompt to guess is the predecessor is higher or lower in value.

Exercise two resembles popular game "Battle Ship". it consists of players finding the "hidden treasures " previously hidden by the first player, in the form of co-ordinates and lastly a classic game of tic tac toe.

“High-Low Guessing” Game

The foundation of any project is first built on planning. Though less exciting and practical it proved effective when author attempted to jump straight in without properly assessing the situation, essentially it is time which is exhausted when a structure hasn't been paved out, the frustrating process is eradicated when a wing it attitude is adopted.

Planning

A clear specification list was created and adopted from the engineering design and product development process, the author created a table (see figure 1) comprising of functional and requirement specification.

requirements	Functional
• Two players • Numbers must disappear after entry • Clear and readable • Must loop at least 7 times • Accurate results	• Players input their names • Numbers disappear in an animated fashion • Give the players the option to play again after completion • Animated graphics upon winning

Figure 1- specifications

listing the basic requirements of the program and additional functionality allows the program to be split up into two stages, the core foundational functions which adhere to the assignment requirements and additional functions for user comfort. the next stage documents a loose implementation structure (see figure 2)

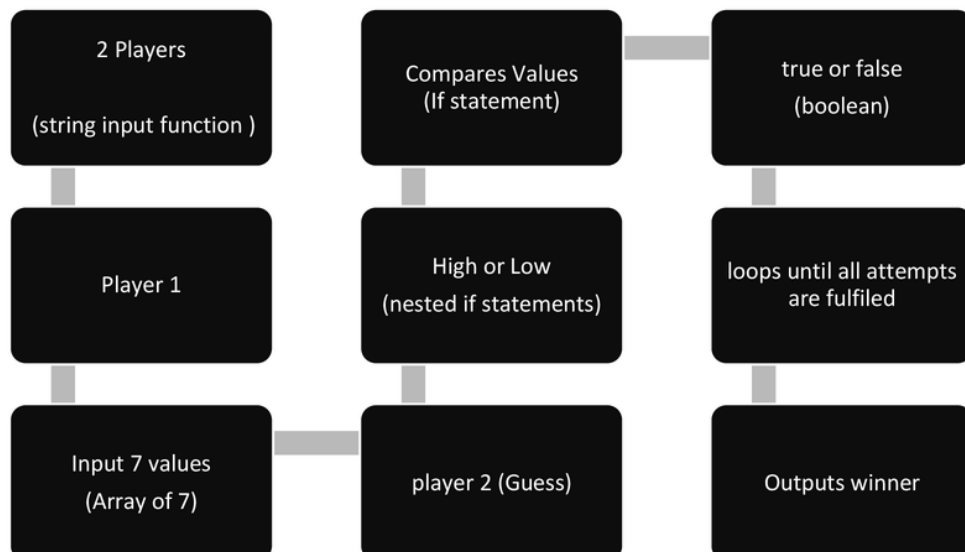


Figure 2-Implementation

Trial and Error

Attempt 1

Figure 2 created a clear achievable structure, it is evident that arrays and loops are a central concept of this code thus creating an array of 7 values, labelled “n1 n2..n7”,(see figure 3) this would allow the user to insert 7 values which would then be compared against player two's values, though it worked, it made the use of an array completely redundant as each value was stored in an integer “n1”

```
cout << player1Name << "  enter 5 numbers between 1 and 10" << endl;

int n1, n2, n3, n4, n5;
getline(cin, input);
stream.str(input);
stream >> n1 >> n2 >> n3 >> n4 >> n5;
stream.clear();

int[5]arrayQ = int arrayQ[5];
int total;
cout << "first number";
getline(cin, input);
stream >> arrayQ[0];

int array[5];
array[0] = n1;
array[1] = n2;
array[2] = n3;
array[3] = n4;
array[4] = n5;
```

Figure 3 -player 1 String Array

In attempt to shorten the code it was changed to “array[5]={n1,n2,n3...}” this also did not work as the index was not consistent with the array datatype. This highlighted the inappropriate use of arrays.

Function “trial()” (see figure 4) attempts to compare both user inputs using a basic if statement, however it was clear via the autos tool that the values were being assigned as the computer's previous memory thus compromising the accuracy of results, additionally the authors attempt to override values “x” and “y” in function “players()” (see figure 5)raised multiple logical fallacies, consequently ceasing to run.

```
void trialH()
{
    getline(cin, input);
    stream.str(input);

    if (x > y && input == "H", x < y && input == "L")
    {
        cout << "correct" << endl;
    }
    else
    {
        cout << "false" << endl;
    }
}
```

Figure 4 – Compares Values

```
players();
array[5] = x;
y = array[1];
trialH();

n3 = x;
n2 = y;
trialH();

n4 = x;
n3 = y;
trialH();

n5 = x;
n4 = y;
trialH();
```

Figure 5 – Overrides Values

Problem Solving

Attempt 2

Understanding the errors with the code allowed the author to implement various problem-solving methods, after careful revision it became clear that the code contained very little logic, a concept paramount for running even the most basic program.

Comparing an integer value to a string value would never be possible and working on the compiler made it difficult to decipher the appropriate remedy thus the author attempted to solve the error using pen and paper. Figure 6 shows the effectiveness of drawing problems out as this consumed half the amount of time it did using the compiler.

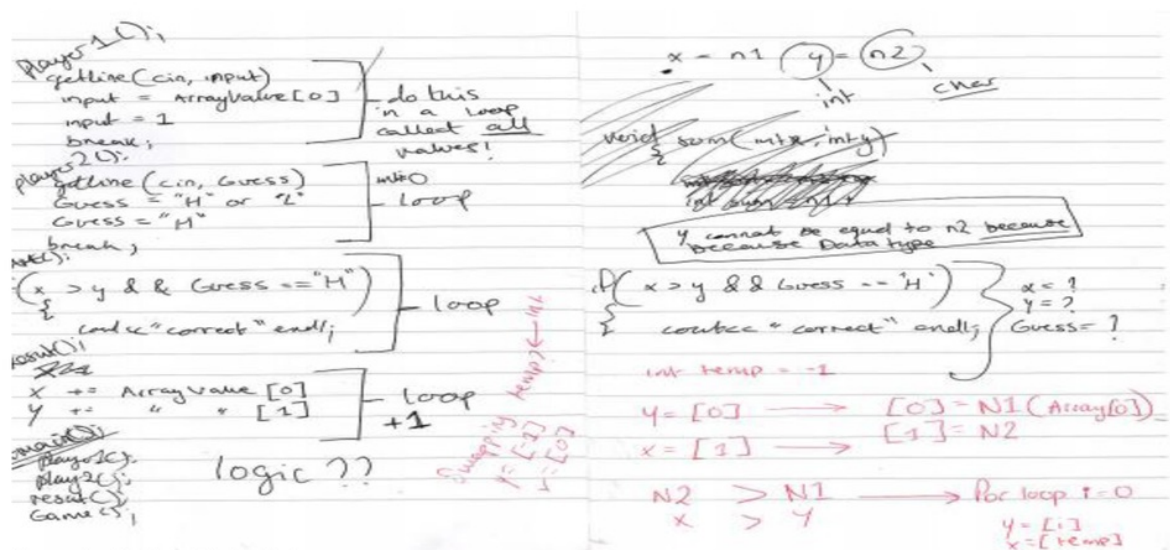


Figure 4 – Problem Solving Notes

Figure 6 displays the physical process of problem solving, establishing a temporary integer meant the array could periodically run through loop without the need of redundant and excessive integers.

Revising past tutorials as a result of failure established a firmer understanding of functions consequently improving the overall hygiene of the code. This also explained the correct use of local and global variables as well as introducing function overloading to reduce excessive code(see figure 7).

```
for (int i = 0; i < 7; i++)
{
    int y = playerInput[temp];
    int x = playerInput[i];
    temp += 1;
    string guess;
    cout << "guess :";
    cin >> guess;

    if (x > y && guess == "H" || x < y && guess == "L")
    {
        cout << " ~~~ correct ~~~ \n" << endl;
        cout << " ~~~ the number was ~~~ " << playerInput[i] << " ~~~ " << endl;
        if (playerInput[i] >= 7)
        {
            cout << "PLAYER TWO WINS" << endl;
        }
    }
    else
    {
        cout << "false " << endl;
        cout << "GAME OVER" << endl;
        cout << "player one WINS" << endl;
        //originally used an if statement, then used i+=8, then used break as its more appropriate.
        break;
    }
}
```

Figure 5- Final implimentation

“Hidden Treasure” Game

Having established working code for the “hi-low game” it became apparent the functions required for achieving desired functionality. Following the same steps as with the first project was an advantage as less time was exhausted. The constant errors faced in the previous game would also be a great advantage as a mental index was created of all possible errors and effective ways to solve them.

Planning

The brief indicated the use of a 2d arrays, a concept the author could not comprehend as it was visually confusing to pin point the exact location of the treasure. There were many attempts to this game none of which involved executing the code on a compiler, instead adopting paper write all functions potentially involved. Then reading the plan systematically until a logical fallacy would appear, this processed was repeated until no possible error could be anticipated.

Trial and Error

Attempt 1

The first attempt consists of Four functions “players”, “treasure array”, “grid” and “player2guess”. The code attempts to jump from one function to the other until the loop index it satisfied. though possible and highly effective, after many attempt the code failed to compile, thus removing “grid” as a function and adding it fully to main(see figure 8).

```
for (int g = 0; g < 3; g++)
{
    players2Input(player2Guess, player2);

    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            if (treasureArray[i][j] == player2Guess[i][j])
            {
                string grid[8][7] = {
                    {"1", "2", "3", "4", "5", "6"},
                    {"1", "2", "3", "4", "5", "6"},
                    {"1", "2", "3", "4", "5", "6"},
                    {"1", "2", "3", "4", "5", "6"},
                    {"1", "2", "3", "4", "5", "6"},
                    {"1", "2", "3", "4", "5", "6"},
                    {"1", "2", "3", "4", "5", "6"},
                    {"1", "2", "3", "4", "5", "6"}
                };
                for (int row = 0; row < 8; row++)
                {
                    for (int column = 0; column < 7; column++)
                    {
                        cout << grid[row][column] << " ";
                        grid[treasureArray[i][j]][treasureArray[i][j]] = { "X" };
                    }
                }
            }
        }
    }
}
```

Figure 6- Grid string Array

Attempt 2

Among other difficulties the most prominent was comparing array "player2Guess" to "treasureArray" then swapping them into the "grid" function so that the value of grid would be overridden by the value in treasure Array and this would only be apparent if the player2Guess was of equal value. This was problematic to implement as player2Guess (see figure 9) was a one-dimensional array whereas the treasure array was two dimensional thus could not be compared.

```
void players2Input(int player2Guess[2])
{
    cout << "\n x: ";
    cin >> player2Guess[0];
    cout << "\n y: ";
    cin >> player2Guess[1];
}
```

*Figure 7- 1D Array***Problem Solving**

The array was remedied by changing player2Guess into a 2D array (see figure 10) this was necessary as it allowed arrays to be compared, Figure 11 shows how having 2d array was professional as it allowed both functions to be compared without adding temporary variables.

```
void players2Input(int player2Guess[1][1], string player2)
{
    cout << player2 << " time to make your guesses using ( x ,y )co-ordinates ";
    string arrayXY[2] = { "x","y" };
    int temp = -1;

    for (int i = 0; i < 2; i++)
    {
        temp += 1;
        cout << arrayXY[temp] << ": ";
        cin >> player2Guess[i][i];
    }
}
```

Figure 8- 2D Array

```
for (int column = 0; column < 7; column++)
{
    cout << grid[row][column] << " ";
    grid[tressureArray[i][j]][tressureArray[i][j]] = { "X" };
}
```

Figure 9-Final Implementation

Findings

The completion of these exercises revealed many concepts, particularly the use of string arrays, after careful consideration it is clear an integer array is considerably more effective as it can be edited in future without adding the strings successively but rather just changing a simple number in the for loop for example figure 13 displays the use of integer arrays for the creation of the grid display.

```

for (int i = 0; i < 9; i++)
{
    if (i < 2)
    {
        cout << "-";
    }
    else if (i > 7)
    {
        cout << i - 1 << "--" << endl;
    }
    else {
        cout << i - 1 << "----";
    }
}

for (int i = 0; i < 6; i++)
{
    for (int j = 0; j < 7; j++)
    {
        if (arrayForGrid[j][i] != 0)
        {
            if (arrayForGrid[j][i] == 1)
            {
                cout << "|_X_|";
            }
            else cout << "|_O_|";
        }
        else cout << "|_|";
    } cout << endl;
}

for (int i = 0; i < 35; i++)
{
    cout << "-";
} cout << endl;

```

Figure 10- Integer Array

Figure 12 is a section of code adapted from an online forum in attempt to complete exercise 3, though the tic tac toe game was complete and working it was not through the authors efforts as all previous attempts failed. However, figure 12 is an indication of how using loops rather than string arrays is better practice especially considering code is supposed to be made to be readable. The inclusion of error input prompts would have been preferable to prevent crashing, a simple while statement such as the following code.

```

While((stream>> x &&stream.eof())!=false)
{
    Cout<<"wrong values, must be between 0-7"<<endl;
    Cout<<"try again"<<endl;
    Stream.clear();
    Getline(cin, input);
}

```

In all three exercises the same structure applied, had the author planned all games together a class would have been created to group all similar functions, using constructors to assure security of values.

As well as noticing functions where similar, after completion the author noticed my functions shared identical result it was only the data type that differed, applying function overload would have meant the author could choose one function and call it to multiply datatype rather than retype the functions and the variables involved.

Conclusion

The progression from the first exercise to the third though designed to get increasingly difficult was the opposite due to properly documenting the errors via call stack and learning from them, by the third I had an index of the typical issues and how to solve them.

A major limitation was lack of knowledge on basic functions such as arrays, this slowed down the process considerably as the author persisted to code even though the code remained faulty, Proficiency was only achieved after revising certain tutorials and references from C++ libraries

Prioritising the tasks between functional and requirements was extremely beneficial as it meant the author spent time on the foundational and basics rather than focussing on the aesthetic of the code and how it will be perceived by the user as this was usually the focal point.

Appendix

Attempt 1: "High-Low Game"

```
#include <iostream>

#include <string>

#include <sstream>

using namespace std;

string input;

stringstream stream;

int players();

int n1, n2, n3, n4, n5;

int x, y;

int array[5];

string player1Name, player2Name;

string result;

void trialH();

// global variables: poor programming practice

int main(int argc, char**argv)
{

    players();

    array[5] = x;

    y = array[1];
```

```

trialH();

n3 = x ;
n2 = y ;

trialH();

n4 = x ;
n3= y;

trialH();

n5=x ;
n4=y ;
trialH();

system("pause");

return 0;
}

int players()
{

cout << "welcome players" << endl;

cout << "player one please enter your name" << endl;
string player1Name;
getline(cin, player1Name);
cout << "player two please enter your name" << endl;
string player2Name;
getline(cin, player2Name);
cout << "rules of the game\n " << endl;
cout << "player one Guesses 5 numbers between 1 and 10 \n" << endl;
cout << "the first guess is displayed \n " << endl;

```

```

cout << "player two must guess if the next number is higher or lower than the one that came before it \n" << endl;
cout << "LETS BEGIN\n" << endl;
cout << player1Name << " enter 5 numbers between 1 and 10" << endl;
int n1, n2, n3, n4, n5;
getline(cin, input);
stream.str(input);
stream >> n1 >> n2 >> n3 >> n4 >> n5;
stream.clear();
int[5]arrayQ = int arrayQ[5];
int total;
cout << "first number";
getline(cin, input);
stream >> arrayQ[0];
int array[5];
array[0] = n1;
array[1] = n2;
array[2] = n3;
array[3] = n4;
array[4] = n5;
cout << "number 1: " << array[0] << "\n" << "number 2: " << array[1];
cout << "\n" << "number 3: " << array[2] << "\n" << "number 4: " << array[3] << "\n" << "number 5: " << array[4] << endl;

// enter loop here for the numbers so you dont have to cout too much
cout << player2Name << " is number 2 lower or higher than number 1: \n make your guess, \n H= higher L = lower "
<< endl;

return 0;
}

void trialH()
{
    getline(cin, input);
    stream.str(input);

    if (x > y && input == "H", x < y && input == "L")
    {

```

```

        cout << "correct" << endl;
    }
    else
    {
        cout << "false" << endl;
    }
}

```

Attempt 2: "High-Low Game"

```
#include <iostream>
```

```
#include <string>
```

```
#include <sstream>
```

```
using namespace std;
```

```
void playersInput()
```

```
{
```

```
    int Player1Input[7];
```

```
    cout << "enter 7 values between 1 and 10" << endl;
```

```
    cin >> Player1Input[0] >> Player1Input[1] >> Player1Input[2] >> Player1Input[3] >> Player1Input[4] >>
    Player1Input[5] >> Player1Input[6];
```

```
    //tried using a for loop here but it didnt work, i think it's because there's too many things in one function.
```

```
    system("CLS");
```

```
    cout << "first number is :" << Player1Input[0] << " " << endl;
```

```
    cout << "\n is player 1's next number higher or lower than the previous" << endl;
```

```
    int temp = -1;
```

```
    cout << " \n **MAKE YOUR GUESS** \n H = HIGHER \n L = LOWER \n" << endl;
```

```

for (int i = 0; i < 7; i++)
{

    int y = Player1Input[temp];
    int x = Player1Input[i];
    temp += 1;
    string guess;
    cout << "guess :";
    cin >> guess;

    if (x > y && guess == "H" || x < y && guess == "L")
    {
        cout << "~~~ correct ~~~ \n" << endl;
        cout << "~~~ the number was " << Player1Input[i] << "~~~ " << endl;

        if (Player1Input[i] >= 7)
        {
            cout << "PLAYER          TWO          WINS" << endl;
        }
    }
    else
    {

        cout << "false " << endl;

        cout << "GAME OVER" << endl;

        cout << "player          one          WINS" << endl;
    }
}

```

appropriate. //originally used an if statement, then used i+=8, then used break as its more

break;

}

}

}

int main(int argc, char**argv)

{

playersInput();

system("pause");

return 0;

}

Final Attempt: "High-Low Game"

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

void playersInput()

{

```

int Player1Input[7];

cout << "enter 7 values between 1 and 10" << endl;

cin >> Player1Input[0] >> Player1Input[1] >> Player1Input[2] >> Player1Input[3] >> Player1Input[4] >>
Player1Input[5] >> Player1Input[6];

//tried using a for loop here but it didn't work, possibly too many things in one function.
system("CLS");

cout << "first number is :" << Player1Input[0] << " " << endl;

cout << "\n is player 1's next number higher or lower than the previous" << endl;

int temp = -1;

cout << " \n **MAKE YOUR GUESS** \n H = HIGHER \n L = LOWER \n" << endl;


    for (int i = 0; i < 7; i++)
    {

        int y = Player1Input[temp];
        int x = Player1Input[i];
        temp += 1;
        string guess;
        cout << "guess :";
        cin >> guess;


        if (x > y && guess == "H" || x < y && guess == "L")
        {
            cout << " ~~~ correct ~~~ \n" << endl;
            cout << "~~~ the number was " << Player1Input[i] << " ~~~ " << endl;


            if (Player1Input[i] >= 7)
            {

```

```

        cout << "PLAYER          TWO          WINS" << endl;
    }

    }
    else
    {

        cout << "false " << endl;

        cout << "GAME OVER" << endl;

        cout << "player          one          WINS" << endl;
        //originally used an if statement, then used i+=8, then used break as its more
appropriate.
        break;
    }

}

}

int main(int argc, char**argv)
{
    playersInput();
    system("pause");
    return 0;
}

```

Attempt 1: "Treasure Hunt Game "

```
#include <iostream>
```

```
#include <string>
```

```
#include <sstream>
```

```
#include <conio.h>
```

```
#include <windows.h>
```

```
using namespace std;
```

```
string playerNames(string name, string playerNum)
```

```
{
    cout << "player " << playerNum << " please enter your name: ";
    cin >> name;
    return name;
}
```

```
void player1Tressure(int tressureArray[3][2], string player1)
```

```
{
    cout << player1 << " hide your THREE tressures using( x ,y )co-ordinates ";
    for (int i = 0; i < 3; i++)
    {
        cout << "\n x: ";
        cin >> tressureArray[i][0];
        cout << "\n y: ";
        cin >> tressureArray[i][1];
    }
}
```

```

        cout << "THANK YOU" << player1 << "YOUR TRESSURES ARE NOW HIDDIN" << endl;

    }

    void players2Input(int player2Guess[2])
    {
        cout << "\n x: ";
        cin >> player2Guess[0];
        cout << "\n y: ";
        cin >> player2Guess[1];
    }

    int main(int argc, char**argv)
    {
        int tressureArray[3][2];
        string player1, player2;
        int player2Guess[2];

        string grid[8][7] = {
            {" ", "1", "2 ", "3", "4", "5", "6\n"},
            {"1", "_", "_", "_", "_", "_", "_ \n"},
            {"2", "_", "_", "_", "_", "_", "_ \n"},
            {"3", "_", "_", "_", "_", "_", "_ \n"},
            {"4", "_", "_", "_", "_", "_", "_ \n"},
            {"5", "_", "_", "_", "_", "_", "_ \n"},
            {"6", "_", "_", "_", "_", "_", "_ \n"},
            {"7", "_", "_", "_", "_", "_", "_ \n"}

        };

        for (int row = 0; row < 8; row++) {

```

```
        for (int column = 0; column < 7; column++) {

            cout << grid[row][column] << "  ";

            grid[i][] = { "x" };

        }

    }

//player names

player1 = playerNames(player1, "one");

player2 = playerNames(player2, "two");

//player 1 treasure locations

player1Tressure(tressureArray, player1);

cout << player2 << " time to make your guesses using ( x ,y )co-ordinates ";

//while(//////)

//{

    players2Input(player2Guess);

    int j = 0;

    for (int i = 0; i > 3; i++)

    {

        if (tressureArray[i][j] == player2Guess[i]&&player2Guess[j])

        {
```

```

        grid[i][j] = { "T" };
        cout << grid[i][j] << endl;
    }

    if (tressureArray[i][j] != player2Guess[i] && player2Guess[j])
    {
        grid[i][j] = { "X" };
        cout << grid[i][j] << endl;
    }
    j = +1;
}

//}

system("pause");
return 0;

}

Attempt 2: "Treasure Hunt Game "
#include <iostream>

#include <string>

#include <sstream>

#include <conio.h>

#include <windows.h>

```

```
using namespace std;
```

```
int main(int argc, char**argv)
```

```
{
```

```
    string grid[8][7] = {
        {" ", "1", "2 ", "3", "4", "5", "6\n"},
        {"1", "_", "_", "_", "_", "_", "_\n"},
        {"2", "_", "_", "_", "_", "_", "_\n"},
        {"3", "_", "_", "_", "_", "_", "_\n"},
        {"4", "_", "_", "_", "_", "_", "_\n"},
        {"5", "_", "_", "_", "_", "_", "_\n"},
        {"6", "_", "_", "_", "_", "_", "_\n"},
        {"7", "_", "_", "_", "_", "_", "_\n"}
    };

    for (int row = 0; row < 8; row++) {
```

```
        for (int column = 0; column < 7; column++) {
```

```
            cout << grid[row][column] << "  ";

        }
```

```
        cout << endl;
    }
```

```

    system("pause");
    return 0;
}

void players()
{
    string player1,player2;
    cout << "player one please enter your name: ";
    cin >> player1;
    cout << endl;
    cout << "player two please enter your name: ";
    cin >> player2;

    system("CLS");

    int x1[3], y1[3];
    cout << player1 << " hide your THREE tressures using( x ,y )co-ordinates ";
    for (int i = 0; i <= 3; i++) {

        cout << "\n x: ";
        cin >> x1[i];
        cout << "\n y: ";
        cin >> y1[i];

    }

    cout << "THANK YOU" << player1 << "YOUR TRESSURES ARE NOW HIDDIN" << endl;

```

```
system("CLS");
```

```
cout<< player2 <<" time to make your guesses using ( x ,y )co-ordinates ";
```

```
int x2[3];
```

```
int y2[3];
```

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

```
    cout << "make your first guess" << endl;
```

```
    cout << "\n x: ";
```

```
    cin >> x2[i];
```

```
    cout << "\n y: ";
```

```
    cin >> y2[i];
```

```
}
```

```
}
```

```
Final : "Treasure Hunt Game "
```

```
#include <iostream>
```

```
#include <string>
```

```
#include <sstream>
```

```
#include <conio.h>
```

```
#include <windows.h>
```

```
// HIDDEN TREASURE GAME
```

```
using namespace std;
```

```
// this function is to register user names
```

```
string playerNames(string name, string playerNum)
```

```
{
    cout << "player " << playerNum << " please enter your name: ";
    cin >> name;
    return name;
}
```

```
void player1Tressure(int tressureArray[3][2], string player1)
```

```
{
    string arrayXY[6] = { "x","y","x","y","x","y" };
    cout << player1 << " hide your THREE tressures using( x ,y )co-ordinates "<<endl;
    int temp = -1;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            temp +=1;
            cout << arrayXY[temp]<<": ";
            cin >> tressureArray[i][j];
        }
    }
    cout << "THANK YOU" << player1 << "YOUR TRESSURES ARE NOW HIDDIN" << endl;
}
```

```
void players2Input(int player2Guess[1][1], string player2)
```

```

{
    cout << player2 << " time to make your guesses using ( x ,y )co-ordinates ";
    string arrayXY[2] = { "x","y"};
    int temp = -1;

    for (int i = 0; i < 2; i++)
    {
        temp += 1;
        cout << arrayXY[temp] << ": ";
        cin >> player2Guess[i][i];
    }

}

int main(int argc, char**argv)
{
    int tressureArray[3][2];
    string player1, player2;
    int player2Guess[1][1];

    player1 = playerNames(player1, "one");
    player2 = playerNames(player2, "two");

    player1Tressure(tressureArray, player1);

    for (int g = 0; g < 3; g++)
    {
        players2Input(player2Guess, player2);
    }
}

```

```

for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 2; j++)
    {
        if (tressureArray[i][j] == player2Guess[i][j])
        {

            string grid[8][7] = {
                {" ", "1", "2 ", "3", "4", "5", "6\n"},
                {"1", "_", "_", "_", "_", "_", "_ \n"},
                {"2", "_", "_", "_", "_", "_", "_ \n"},
                {"3", "_", "_", "_", "_", "_", "_ \n"},
                {"4", "_", "_", "_", "_", "_", "_ \n"},
                {"5", "_", "_", "_", "_", "_", "_ \n"},
                {"6", "_", "_", "_", "_", "_", "_ \n"},
                {"7", "_", "_", "_", "_", "_", "_ \n"}
            };
            for (int row = 0; row < 8; row++)
            {
                for (int column = 0; column < 7; column++)
                {
                    cout << grid[row][column] << " ";
                    grid[tressureArray[i][j]][tressureArray[i][j]] = { "X" };
                }
            }
        }
    }
}

```

```
    system("pause");  
    return 0;  
}
```

Software Development

GRADEMARK REPORT

FINAL GRADE

63/100

GENERAL COMMENTS

Instructor

Exercise 1 (50) 44

Exercise 2 (30) 14

Exercise 3 (20) 5

A reasonable attempt made - exercise 1 works well. A fair attempt made at exercise 2, though the functionality isn't quite as intended and exercise 3 is incomplete. The code is a little untidy, but still a Good attempt made generally.

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
