Project 1

<Mastermind>

CIS-7- 43527

Name: Koksal, Attila

Date: 4/15/2022

Introduction

This project revolves around the game, Mastermind. I am doing this game since it's a game that is really enjoying and fun to play.

Summary

The program is 185 lines of code. This project meets the criteria of the first project because of how it implements a simple version of the game Mastermind in C++ code. This project utilizes pointers, arrays, and many more as well as 13 variables are used in this project. 32 constructs were utilized in this project. It was challenging for me, especially to be able to provide the user the hints to help with guessing the correct answer when the user got a guess wrong since it used the user's previous input to construct the hints provided thereafter. It took approximately 15 hours, including research, review, debugging, and coding, in total to complete this project.

Description

With the goal of creating a playable mastermind game, there are various sections and challenges to be completed. These steps include using a random number generator to generate a random correct answer for the user to try to guess at for a designated number of guesses depending on the chosen difficulty level. Then, enable the user to input possible combinations to try to get the correct answer through the usage of hints, which are derived from the user's previous inputs in order to assist the user in getting the correct answer and winning the game. The hints by far is the most complex part in this entire game but also the most interesting since the user can use those to increase his/her chance at winning the game. At the end of the # of total guesses available depending on the selected difficulty, if the user doesn't guess within that range, they lose whereas if they are able to guess then they are proclaimed the winner. The user can play the game for as long as he/she wants to unless provided with a "no," which thereby the program concludes to an end.

Pseudocode

```
Select the difficulty level of Mastermind to play
While game is not over
    Enter a guess
        If the guess is correct
            Print player win
            If player wants to play game again
                Continue game
            End the game
        If the guess is incorrect
```

Print the number of pegs with the correct color and position

Print the number of pegs with the correct color, but in the wrong position

Notify user how many guesses are left to use

Ask user to enter another guess

Repeat steps above until either the user loses or wins the game

Variables

string choice = “”;

- The name of the variable is called choice.
- Its type is a string.
- It stores which difficulty level the player chooses to play the game in.
- It is located on line 23.

int guesses;

- The name of the variable is called guesses.
- Its type is an integer.
- It stores the number of guesses available at each difficulty level.
- It is located on line 36.

char answer [8];

- The name of the variable is called answer.
- Its type is a character array holding 4 elements.
- It stores the randomly generated correct answer for the game.
- It is located on line 50.

bool repeat = true;

- The name of the variable is called repeat.
- Its type is a boolean.
- It determines how long the player wants to play the game for.
- It is located on line 51.

int numBlack = 0;

- The name of the variable is called numBlack.
- Its type is an integer.
- It represents the # of black pegs.
- It is located on line 53.

int numWhite = 0;

- The name of the variable is called numWhite.
- Its type is an integer.

- It represents the # of white pegs.
- It is located on line 54.

string guess;

- The name of the variable is called guess.
- Its type is a string.
- It stores the guess given by the user as input for the game to be played.
- It is located on line 68.

bool correct = false;

- The name of the variable is called correct.
- Its type is a boolean.
- It sets the correct answer automatically to false before any input is taken.
- It is located on line 69.

int z = 0;

- The name of the variable is called z.
- Its type is an integer.
- It acts as a placeholder for the game code to run.
- It is located on line 70.

string repeatagain = “”;

- The name of the variable is called repeatagain.
- Its type is a string.
- It stores the user’s input for if they want to play the game again.
- It is located on line 102.

bool exact = true;

- The name of the variable is called exact.
- Its type is a boolean.
- It checks in assumption that the given input guess is in fact correct.
- It is located on line 139.

int rightColorAndPlacement;

- The name of the variable is called rightColorAndPlacement.
- Its type is an integer.
- It keeps a total of # of pegs that are right color/right spot to use in the hints section.
- It is located on line 140.

int rightColorWrongSpot;

- The name of the variable is rightColorWrongSpot.
- Its type is an integer.
- It keeps a total of # of pegs that are right color/wrong spot to use in the hints section.

- It is located on line 141.

int wrongWhiteGuessed = 0;

- The name of the variable is called wrongWhiteGuessed.
- Its type is an integer.
- It tracks the # of incorrectly-placed pegs that are white to use in the second hint.
- It is located on line 143.

int wrongBlackGuessed = 0;

- The name of the variable is called wrongBlackGuessed.
- Its type is an integer.
- It tracks the # of incorrectly-placed pegs that are black to use in the second hint.
- It is located on line 144.

Program

```
// Mastermind

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>


using namespace std;


bool validGuess(string);//determines if a player's guess is even


string capitalize(string);//used to make every letter uppercase in an input string


bool hintCreator(string, char*, int, int);//creates the hints for the game and checks if user has
guessed the correct combination




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

   string choice = "";
```

```
cout << "Welcome to Mastermind." << endl;

cout << "Please select a difficulty level:" << endl;
cout << "1 = Easy. 2 = Medium. 3 = Hard." << endl;//Select a difficulty
cin >> choice;

while (choice != "1" && choice != "2" && choice != "3"){//ask again if choice isn't valid
    cout << "Invalid choice. Please enter a 1, 2, or 3: ";
    cin >> choice;
}

int guesses;
if (choice == "1"){
    guesses = 12;
    cout << "Easy selected!" << endl;
}
else if (choice == "2"){
    guesses = 10;
    cout << "Medium selected!" << endl;
}
else{
    guesses = 8;
    cout << "Hard selected!" << endl;
}//the difficulty level decides how many guesses the player gets

char answer [8];//initialize a character array of length 4
bool repeat = true;//determine how long player wants to play the game
do{
    int numBlack = 0;//# of black pegs
    int numWhite = 0;//# of white pegs
    srand(time(0));//set up a time using srand()

    for (int i = 0; i < 8; i++){//for each character
        if(rand() % 2 == 0){//generate a random number and then determine if value is odd or
even
            answer[i] = 'W';//randomly assign every peg to be white or black
            numWhite++;
        }
        else{
            answer[i] = 'B';
            numBlack++;
        }
    }

    string guess;
    bool correct = false;
    int z = 0;
    while (z < guesses && !correct){//game code
        cout << "Enter a guess: ";//ask for a guess from the user
        cin >> guess;
        while (!validGuess(guess)){//Once a guess is made, validate it and if it's not valid, keep
asking until one that qualifies is received
            cout << "Invalid input. Please enter a four-character input consisting solely of B and
W" << endl;
            cout << "Enter a guess: ";
            cin >> guess;
        }
```

```
            correct = hintCreator(capitalize(guess), answer, numWhite, numBlack);//create hints for
the given guess and determine if guessed correctly


            if(correct){//if user is correct, they win the game
                cout << "Congratulations, you win!" << endl;
            }
            else{//if user is wrong, notify user of many guesses they have left
                cout << "Incorrect! You have " << guesses-z << " guesses left!" << endl;
            }


            z++;//increment z
        }


        if(!correct){//if number of guesses is exceeded, the player loses
            cout << "You're all out of guesses! I win!" << endl;
            cout << "The code was ";
            for (int q = 0; q < 8; q++){//print the actual answer to the game
                    cout << answer[q];
            }
            cout << "!" << endl;
        }


        cout << "Play again? (y/n) "; //ask the user if they want to play again after the game has
concluded
        string repeatagain = "";
        cin >> repeatagain;
        while(repeatagain != "y" && repeatagain != "Y" && repeatagain != "n" && repeatagain != 
"N"){//validate the user's given input
            cout << "Invalid input. Enter either 'y' or 'n': ";
            cin >> repeatagain;
        }

        if(repeatagain == "n" || repeatagain == "N"){
            repeat = false;
        }//if user doesn't want to play again, the do-while loop breaks and if user does want to play 
again, just let the loop continue as it is

    }while(repeat);
}
bool validGuess(string guess){//determines if a player's guess is even
    if(guess.length() != 8){//if player inputs a four character long guess, then don't accept it
        return false;
    }
    for(int i = 0; i < 8; i++){//if guess has four letters, then go through the input and check if each 
letter is either a w, W, b, or B
        if(toupper(guess[i]) != 'W' && toupper(guess[i]) != 'B'){
            return false;//if input is wrong at all, then invalid guess
        }
    }

    return true;//if true still, then valid guess from player

}

string capitalize(string s){//used to make every letter uppercase in an input string
```

```
    for(int i = 0; i < s.length(); i++){
        s[i] = toupper(s[i]);
    }

    return s;
}

bool hintCreator(string guess, char* answer, int numW, int numB){//creates the hints for the
game and checks if user has guessed the correct combination
    bool exact = true;//the assumption that the guess is in fact correct
    int rightColorAndPlacement = 0;//keeps totals of # of pegs that are right color/right spot and
right color/wrong spot to use in the hints section
    int rightColorWrongSpot = 0;

    int wrongWhiteGuessed = 0;
    int wrongBlackGuessed = 0;//tracks the # of incorrectly-placed pegs that are white and black
to use in the second hint
    //
    for(int i = 0; i < 8; i++){//for every letter of the given guess
        if(guess[i] == answer[i]){//check to see if it matches with the corresponding peg in the
given guess
            rightColorAndPlacement++;//increment the number of perfectly right pegs
            if(guess[i] == 'W'){//keeps track of if the perfectly correct peg is white or black and this
comes into account when providing the second hint
                numW--;
            }
            else if(guess[i]== 'B'){
                numB--;
            }
        }
        else{//if the peg in the given guess and the corresponding peg in the correct answer are not
the same
            exact = false;//if so, change the value of exact
            if(guess[i] == 'W' ){//record an incorrect peg either white or black in the two counter
variables below
                wrongWhiteGuessed++;
            }
            else if(guess[i]== 'B' ){
                wrongBlackGuessed++;
            }
        }
    }

    if (wrongWhiteGuessed >= numW) {
        rightColorWrongSpot += numW;
    }
    else {
        rightColorWrongSpot += wrongWhiteGuessed;
    }

    if (wrongBlackGuessed >= numB) {
        rightColorWrongSpot += numB;
    }
    else {
        rightColorWrongSpot += wrongBlackGuessed;
    }

    cout << "You have " << rightColorAndPlacement << " pegs of the correct color and position."
<< endl;
```

```
    cout << "You have " << rightColorWrongSpot << " pegs that are the right color, but in the wrong position." << endl << endl;


    return exact;//returns if player guess is the exact answer
}
```