Richard Aguilar

Prof. Lehr

18 April 2021

Mastermind Write-up

The mastermind game is played by calling the masterMind() function. The master idea behind the function was to ask two questions to set the gamer for the user. The first question will ask for the number of pegs they want to play with and can choose from 4 and 8. The second question ask whether the player wants duplicates to be allowed in the game. This means that with duplicates off the code generated will not have any duplicated and user may not enter duplicated into their guess.

```
cout<<"Please enter the a size of pegs you want to play with from 4 and 8"<<endl;
do{
    cout<<"SIZE: ";
    cin>>SIZE;
    if(SIZE>8||SIZE<4){ cout<<"INVALID: INPUT OUT OF BOUNDS"<<endl; }
}while(SIZE>8||SIZE<4);
```

The above code shows the do-while block of code where the first question is asked. The reason we use a do-while loop is to prevent invalid inputs. When the user inputs a size out of bounds the repeats the question and spits out an error message to enter a valid input.

```
//creates the string that result has to equal to win the game
string win_result;
for(int i = 0; i < SIZE; i++){
    win_result += '+';
}
```

In this code block the I initialize a string called ‘win_result’ which will store what the winning result should look like based on the input of the user. For example, if the user enters a size of 4 the winning result will be “++++” or if they enter 8 it will be “++++++++”.

```
string result;
for(int i = 0; i < SIZE; i++){
    result += '_';
}
```

The above code block is similar to the code block that generates the winning result except is to generate an empty result string filling it dynamically with '_' which represents empty spaces

```
› createCode(code, SIZE, dupe);
```

Here I call a function I created which takes in the code array, size, and dupe as parameters. The code will work depending on whether the user enabled duplicates or not.

```
void createCode(int code[], int SIZE, int dupe){
    srand(static_cast<unsigned int>(time(0)));
    if(dupe == 0){
        for(int i = 0; i < SIZE; i++){
            code[i] = (rand()%8)+1;
            while(inArray(code, code[i], i)){
                code[i] = (rand()%8)+1;
            }
        }
    }else{
        for(int i = 0; i < SIZE; i++){
            code[i] = (rand()%8)+1;
        }
    }
}
```

Above is the createCode() function that takes the code array, size, and dupe as parameters. First to generate a random code the function initialized a srand() seed. Then it checks whether duplicates are allowed or not. If duplicate ARE NOT allowed then it enters a for loop which helps create a random integer for each element in the code array after that the code checks

whether the number has been inputted in the array already by calling the inArray() functions and chooses another random number. If duplicates ARE allowed then it will enter random numbers disregarding whether they are duplicates.

```
short turns = 0;
while(turns != 10 && result != win_result){
    cout<<"=========TURN: "<<turns+1<<"==========="<<endl;
    enterGuess(guess, SIZE, dupe);
    turns++;
    printCode(guess, SIZE);
    result = getResult(code, guess, SIZE);
    cout<<"Result: " << result << endl;
}
if(result == win_result){
    cout<<"CONGRATS YOU WON!!"<<endl;
    cout<<"YOU WON IN " << turns << " TURNS"<< endl;
    cout<<"The winning code is: ";
    printCode(code, SIZE);
}else{
    cout<<"YOU LOST :( "<<endl;
    cout<<"The correct code is: ";
    printCode(code, SIZE);
}
```

In the above code is the heart of the game. I initialize a turns variable that will increment each turn in the game. I used a while-loop to loop around. In this case I could have used a for-loop to achieve the same thing, but I chose a while loop because it made it easier to look at the conditional statement. During the while-loop the user enter their guess by calling the enterGuess() function. Next the turns variable increments and the user inputted code is also printed out. The result variable is then equal to the getResult() function that will be mentions later. This repeat until either 10 turns has passed, or the result is equal to the win result. After

the based on whether you won or not you either get a congratulation message or a you lost message.

```
bool inArray(int code[], int num, int SIZE){
    for(int i = 0; i < SIZE; i++){
        if(code[i] == num){ return true; }
    }

    return false;
}
```

This is the inArray function that takes in the code array, the number they are checking and the size. The purpose of this function is check if a number that either user inputted or the program generated. It does this by using a for loop and checking each number up too the most recent number if there is a number similar to the new number. If so the function returns true. If not, then it will return false.

```
void enterGuess(int guess[], int SIZE, int dupe){

    for(int i = 0; i < SIZE; i++){
        if(dupe == 0){
            cout<<"Please enter numbers between 1 and 8 no duplicates"<<endl;
            cout<<"Please enter hole #"<<i+1<<':';
            cin>>guess[i];
            while(guess[i] < 1 || guess[i] > 8 || inArray(guess, guess[i], i)){
                cout<<"ERROR: INVALID NUMBER INPUTTED"<<endl;
                cout<<"PLEASE DO NOT ENTER DUPLICATES OR ANYTHING LESS THAN 1 OR GREATER THAN 8"<<endl;
                cout<<"Please enter hole #"<<i+1<<':';
                cin>>guess[i];
            }
        }else{
            cout<<"Please enter numbers between 1 and 8"<<endl;
            cout<<"Please enter hole #"<<i+1<<':';
            cin>>guess[i];
            while(guess[i] < 1 || guess[i] > 8 || inArray(guess, guess[i], i)){
                cout<<"ERROR: INVALID NUMBER INPUTTED"<<endl;
                cout<<"PLEASE DO NOT ENTER ANYTHING LESS THAN 1 OR GREATER THAN 8"<<endl;
                cout<<"Please enter hole #"<<i+1<<':';
                cin>>guess[i];
            }
        }
    }
}
```

The enterGuess function takes in the array guess, size, and dupe as parameters. Depending on whether you enabled duplicated you will get a different response. For example, if

you did not allow duplicated you enter a number for your guess and after each guess it enters a conditional statement. The purpose of this while-loop is to check for invalid inputs. The conditional statement checks whether the inputted number is larger than 8 or less than 1 or if the number has been inputted in the array already. If any of the conditions are met then the it will spit an error and ask to input a new valid number.

```
string getResult(int code[], int guess[], int SIZE){
    unordered_map<int, int> freq;
    char result[SIZE];
    string str;
    for(int i = 0; i < SIZE; i++){
        freq[code[i]]++;
        result[i] += '_';
    }

    int arr_pos = 0;
    for(int i = 0; i < SIZE; i++){
        if(freq[guess[i]] > 0){
            result[arr_pos] = '-';
            arr_pos++;
            freq[guess[i]]--;
        }
    }

    arr_pos = 0;
    for(int i = 0; i < SIZE; i++){
        if(guess[i] == code[i]){
            result[arr_pos] = '+';
            arr_pos++;
        }
    }
    for(int i = 0; i < SIZE; i++){
        str += result[i];
    }
    cout<<endl;
    return str;
}
```

The above code is the getResult() functions that takes the code array, the guess array, and the size of the arrays as parameters. I used a unordered map to count the frequency of each number of the code array. I chose to use a unordered map because I found it better than using a nested for loop inside another for loop. I initialize a array of chars named result and a string named str. The variable arr_pos is created to keep track of the result array as it traverses different for loops. The first for loop iterated through each element looking whether its in the frequency map. If a number in the guess is also found in the code then it sets result [arr_pos] equal too '-' meaning there's a correct number but incorrect position. The second for-loop checks if any number in the code is the same number and exact position if so it will be equal too '+'. The final for loop concatenates each element in the array into a string an returns it.