INDEX:

INTRODUCTION:

In this project we are creating a program that stimulates Classic Card game. In which there are two players. The game is about user drawing each card from a deck of cards. Maximum rank card wins.

A winner is decided in the form of the best of three rounds. Before the game start we are shown which number of round is being played and after that we are displayed Player One is playing. After that Player one will be asked to provide any number from 1-52. After Player one turn Player second goes with the same process after that Winner of the round is declared and Round 2 will start following the same previous process.

After playing 3 rounds game winner will be declared by comparing the result of 3 rounds. And Scoreboard is also displayed of both the players.

Many different sources have been used to make this project, the reason are some problems arisen while making the project. What has been used is both books and internet to get a good explanation and optimized way to approach certain problems.

PROBLEM DESCRIPTION:

CREATING FUNCTIONS FOR THE PROGRAM:

We need to make 3 functions for taking Data from user, responding to the Data taken from user, shuffling the cards and returning those values inside the main function.

In this assignment there is a need to divide the code into smaller pieces of code and the reason is because the code becomes hard to read without doing that. When there is a code that needs to be repeated in different parts of the program it makes the code longer and harder to read and it can also lead to different types of complications within the program.

USING FUNCTIONS TO BREAK CODE IN SMALLER PARTS:

This part of the problem is about the discussion about the Functions and how the make our custom functions for our required problem. It make our code easier to work, reduce complications and easy to read the code and understand it. It just like making departments for each type of work to handle them separately.

RETURNING THE DATA:

Since every function has done their work now we will like to assemble all the data to the final

Part of the program. It's like after making other parts are made now we want to make a final product by assembling all the parts for the final product.

SOLUTION DESIGN:

We will start from the Function which take User Input so that we can start our program according to the user Input.

FUNCTION: int UserInput()

We will make int UserInput Function we are taking int as Datatype of the function because we need to send value as well as well want to get some data outside from the function.

The function start with asking Enter the card number form 1-52. After taking that Userinput we need to store it in variable int Input and we will return the input data by decrementing it by 1.

```
int UserInput(){
    cout << "ENTER YOUR CARD NUMBER FROM 1-52: ";
    int Input;
    cin >> Input;
    return Input-1;
}
```

One question may arise why we are returning value after decrementing it by 1. It is because we are storing the Deck in Two dimensional Array and we know that Array starts from 0 so we need to take the first Card from the deck if some Input 1 it returns 0 since 0 is the first position in array.

FUNCTION: pair<string,string>Response()

We will make function that consist of 2D Array which store both the Rank of the Cards and the Suit of the Cards inside 1 Array. We could have done it using Normal array but we will have to take 2 array for creating it. So as per good programming Habit we will take 2D Array.

We will display which card is been draw by which player and we will fetch the data from CreateDeck[] Array were we have made a Deck it fetch Rank from the Array-> second dimension and we need to fetch Suit from the Array->First dimension.

And we need to return the Card value.

```
pair<string, string> Response(int Input,pair<string, string>
CreateDeck[]){
    cout << "HE DRAWS: " << CreateDeck[Input].second << " OF " <<
CreateDeck[Input].first << "\n";
    return CreateDeck[Input];
}
```

FUNCTION: void Shuffle()

We will make a Function the do Shuffle part of the code. So that our Card positions and Suits position get randomly changed so that we get unique value each time. So that program remain fair to anyone who plays the game.

Since this is a void function we don’t expect any return values from this.

```
void Shuffle(pair<string, string> CreatDeck[]){
   random_shuffle(CreatDeck, CreatDeck+52);
}
```

MAIN FUNCTION:

After all the functions are ready we need to call them inside the main function to actually see them working. So inside main function we will first use srand Function to make sure every time

The code runs the randomshuffle function doesn’t repeat itself and It gives completely random values each time.

Then we will call our 2D Array which is pair<string, string> Deck[52] and we will populate our array with suit[] values and card[] values.

```
   pair<string, string> Deck[52];
   string Suit[] = {"Spade","Heart","Diamond","Club"};
   string Card[] = {"A","2","3","4","5","6","7","8","9","10"
,"J","Q","K"};
```

We will Initialize EachPlayerScore so that we can store the score inside that variables.

```
int PlayerScore1 = 0, PlayerScore2 = 0;
```

Now here we come to main part of the code we will initialize a for loop which run for 3 times for one Game to choose the winner.

We will start the game with displaying Number of Round and call the shuffle function and pass the deck values inside the function.

After that we display which Player is playing and then we will call the Input Function and store the return value inside a variable called Input1/2 and then we will be passing the value of input variable inside 2D pair< string, string > and we will call Response functions(Input, Deck);

```
    int Input1 = UserInput();
    pair<string, string> Response1 = Response(Input1,Deck);

    cout << "PLAYER SECOND PLAYING: " << endl;
    int Input2 = UserInput();
    pair<string, string> Response2 = Response(Input2,Deck);
```

We also need to initialize PlayerCard1/2, PlayerSuit1/2 for both the players.

```
int PlayerCard1 = -1, PlayerCard2 = -1;
int PlayerSuit1 = -1, PlayerSuit2 = -1;
```

We need to set the values of PlayerCard1/2 and PlayerSuit1/2 and We need to initialize for loop which repeat itself for 4 time because there are 4 Suit "Spade","Heart","Diamond","Club"

```
    for(int j=0; j<4; j++){
        if(Response1.first == Suit[j]){
            PlayerSuit1 = j;
        }
        if(Response2.first == Suit[j]){
            PlayerSuit2 = j;
        }
    }
```

We also need to initialize for loop which repeat itself for 13 times because there are 13 Card which are

"A","2","3","4","5","6","7","8","9","10","J","Q","K"

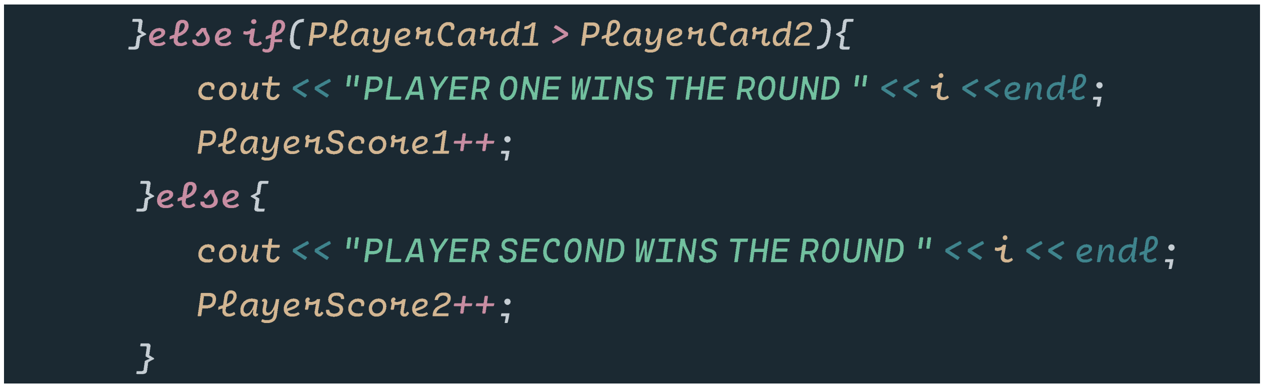
```
    for(int j=0; j<12; j++){
        if(Response1.second == Card[j]){
            PlayerCard1 = j;
        }
        if(Response2.second == Card[j]){
            PlayerCard2 = j;
        }
    }
```

COMPARING INPUTS FOR DECLARING THE WINNERS:

If Player1 and Player 2 both gets the same cards we need to check whose suits is greater. One who has the Greater Suit card wins the round and we increment the score system by 1.

```
    if(PlayerCard1 == PlayerCard2){
        if(PlayerSuit1 < PlayerSuit2){
            cout << "PLAYER ONE WINS THE ROUND " << i <<endl;
        }else{
            cout << "PLAYER SECOND WINS THE ROUND " << i <<endl;
        }
```

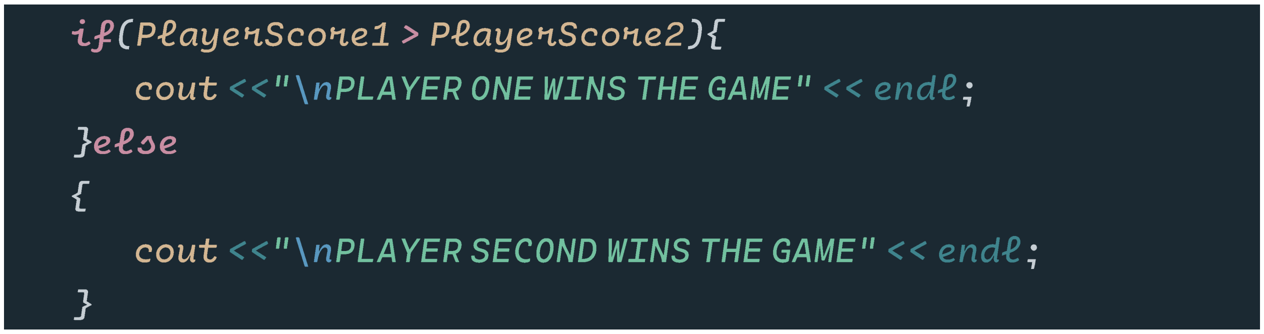
In case Cards are different we need to compare both the players' card and after comparing we declare the round winner who has greater card and we again need to increment the score system by 1.



```
    }else if(PlayerCard1 > PlayerCard2){
        cout << "PLAYER ONE WINS THE ROUND " << i <<endl;
        PlayerScore1++;
    }else {
        cout << "PLAYER SECOND WINS THE ROUND " << i << endl;
        PlayerScore2++;
    }
```

DECLARING GAME WINNER:

We simply compare the Playerscore of both the players whoever has the more score we need to declare him as a winner



```
  if(PlayerScore1 > PlayerScore2){
      cout <<"\nPLAYER ONE WINS THE GAME" << endl;
  }else
  {
      cout <<"\nPLAYER SECOND WINS THE GAME" << endl;
  }
```

Reference

REEMA THAREJA PROGRAMING WITH CPP

Christian, Lennerholt " Videoföreläsningar" (2020) https://his.instructure.com/courses/3628/pages/videoforelasningar?module_item_id=33727

Mike, Dane “ C++ Tutorial for Beginners - Full course” (24 -Aug -2018)

https://www.youtube.com/watch?v=vLnPwxZdW4Y

Skolverket “ Kort vetenskaplig rapport “ (2020)

https://www.skolverket.se/download/18.1d7693d81684bec928273a/1548151697022/kort-vet

enskaplig-rapport-naturvetenskap-teknik-grundskola.pdf

Marshall, Gunnell “ How to create a flowchart “ (11-June-2019)

https://www.howtogeek.com/424397/how-to-create-a-flowchart-in-word/

Jimmy, Wales “Procedural programming” (15- Sep-2020)

https://en.wikipedia.org/wiki/Procedural_programming

Simon K Jensen “ C++ på riktigt “ (25-Sep-2019)