**Blackjack/21**

**Card Game**

Drake Gerger

CIS 5

Fall 2017

47471

**Table of Contents**

**Introduction 3**

**How the Card Game Works 3**

Object of the Game 3

Rules of the Game 3

How to Win BlackJack/21 4

**My Approach to the Game 4**

Translating Game Play Rules to Programming Language 4

Similarities to the Card Game 4

Differences from the Card Game 5

**The Logic of it All 5**

Flowchart 5

Pseudo Code 7

**Constructs & Concepts Utilized 8**

Iostream Library 8

Cstdlib Library 8

cTime 8

Data Types 8

Conditional Statements 9

Loops 9

**Proof of a Working Product 9**

**References 11**

**Program 12**

**Introduction**

“Blackjack is very scientific. There's always a right answer and a wrong answer. Do you take a card, increase your bet, bet big or bet small. There's absolutely a right and wrong answer.”

* Charlie Ergen

BlackJack, also referred to by younger kids as 21, is a popular single player game that can be played at the same table with other people. The game is mostly played by people 18 and older and usually involves some type of bet in ordered to be played. The game is also played by younger kids in which they play for fun and they call it 21. BlackJack can also serve as awesome classic game that teaches you how to count and add their cards numbers. All the players are required to do is to have their cards displayed and to count them to make sure they do not go over 21 or else they lose. Although this game does required a little bit of logic to think and make a decision, the game is still a lot fun and competitive. The process of having cards randomly dealt to the player and the house having an increased chance to win is what really spoke to me when thinking and creating a card game.

**How the Card Game Works**

**Object of the Game**

Get 21 points on the player's first two cards without a dealer blackjack, reach a final score higher than the dealer without exceeding 21, or let the dealer draw additional cards until their hand exceeds 21.

**Rules of the Game**

Blackjack is a single player game but when at a casino, blackjack is between 2 and 7 people usually at one table. The game is very simple:

1. Shuffle the correct amount of cards based on players. For an example if the dealer has 7 players, the dealer would most likely have to shuffle around 416 cards. Once shuffled then the dealer deals 2 cards to the player or players while the first card dealt is face up or down depending on the casino and the second card is faced up. Therefore, each player should have a one card face down or up and second card up at all times. Jokers are not used in this game.
2. Players should then add up their cards and make a decision whether to “Hit” or “Stay” based on how much your cards add up too. Who ever has 21 or the close to 21 wins. There can also be a tie in which case no one wins or loses. The ranks of cards are as follows:
   * All number cards are valued according to their number.
   * Of the face cards Joker, Queen, and King have a value of 10. The Ace has a value of 11 or 1. For an example, if the house or player receives a King and a Ace that is 21 but if the house or player receives a 5 and another 5 and then a King and then receives a Ace, that Ace can be considered a 1.
3. Keep “Hitting” until you the player has cards that equal to 21 or close to it. You the player can also “Stay” and wait to see if anyone/dealer has 21 or lower.

**How to Win Blackjack/21**

Players win by not busting and having a total that is higher than the dealer. Players could also win by blackjack without the dealer getting a blackjack. If the player and dealer have the same total not counting blackjacks, this is called a "push", and the player typically does not win or lose money on that hand. If the player bust or does not have a high enough total the dealer wins.

**My Approach to the Game**

**Translating Game Play Rules to Programming Language**

While thinking of how I was going to program this awesome card game, a few questions surfaced:

* “Since the Blackjack game has face cards with a value of 10 and even one face card with 2 values, how do I code face cards with values?”
* “Should I have the computer tell the player how much their hand is or should I let them add the numbers and/or face cards.”
* “How will the player win or lose?”
* “How will the dealer win by default if the player has too much of a value?”

After long hours of planning my program and messing with the above questions. I decided to read the book from Gaddis and look at GitHub from Professor Lehr’s repository. Most of my questions involved loops and functions and some arrays. Therefore, I had to come up with the code using different parts of other projects and pulling in some other resources.

**Similarities to the Card Game**

My Blackjack/21 program follows similar rules of play as the table card game:

* The player and the dealer have 2 cards, and then it is determined who wants to “Hit” or “Stay”.
* If the players cards equal 21 it's a immediate win unless more players.
  + If the players cards equal more than 21 the house’s cards win even if they were as low as 2 (Ace & Ace).

**Differences from the Card Game**

The main difference from the card game is the player’s card count and the house’s card count. I decided that it would be beneficial to the player if they didn’t have to sit their and add up the cards. I decided to add a card count next to each player/dealer name.

* Before pressing “H” or “S” it will tell the player his/hers card count and what it is summed too.
* After you press “H” it will show the dealer's hand to hopefully make assist the player in making a strategic decision.

Since the players cards don’t have a digital number next to each card in the actual table game. I assigned a total value to each time a player “Hits”. This total value is a running total of what your cards will equal.

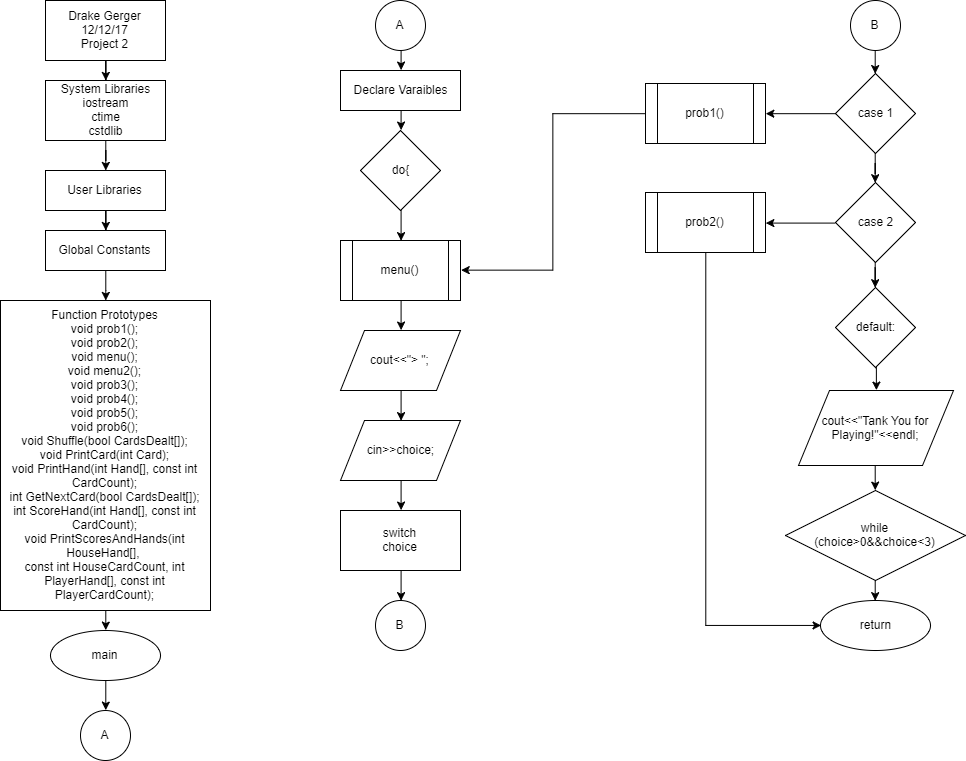
Lastly, a player wins or loses based on their score when they “bust”, “stay”, or “push”. If the player at any time during the game, can exit the game and the computer will tell them “Thank You for Playing”.

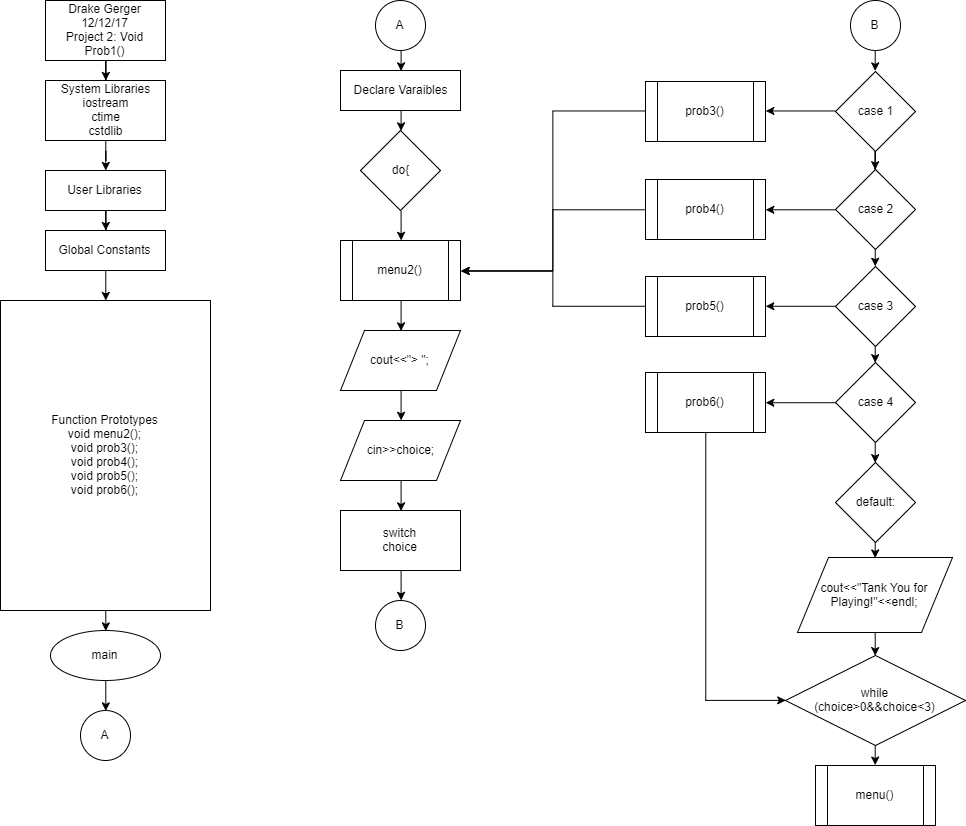
**The Logic of it All**

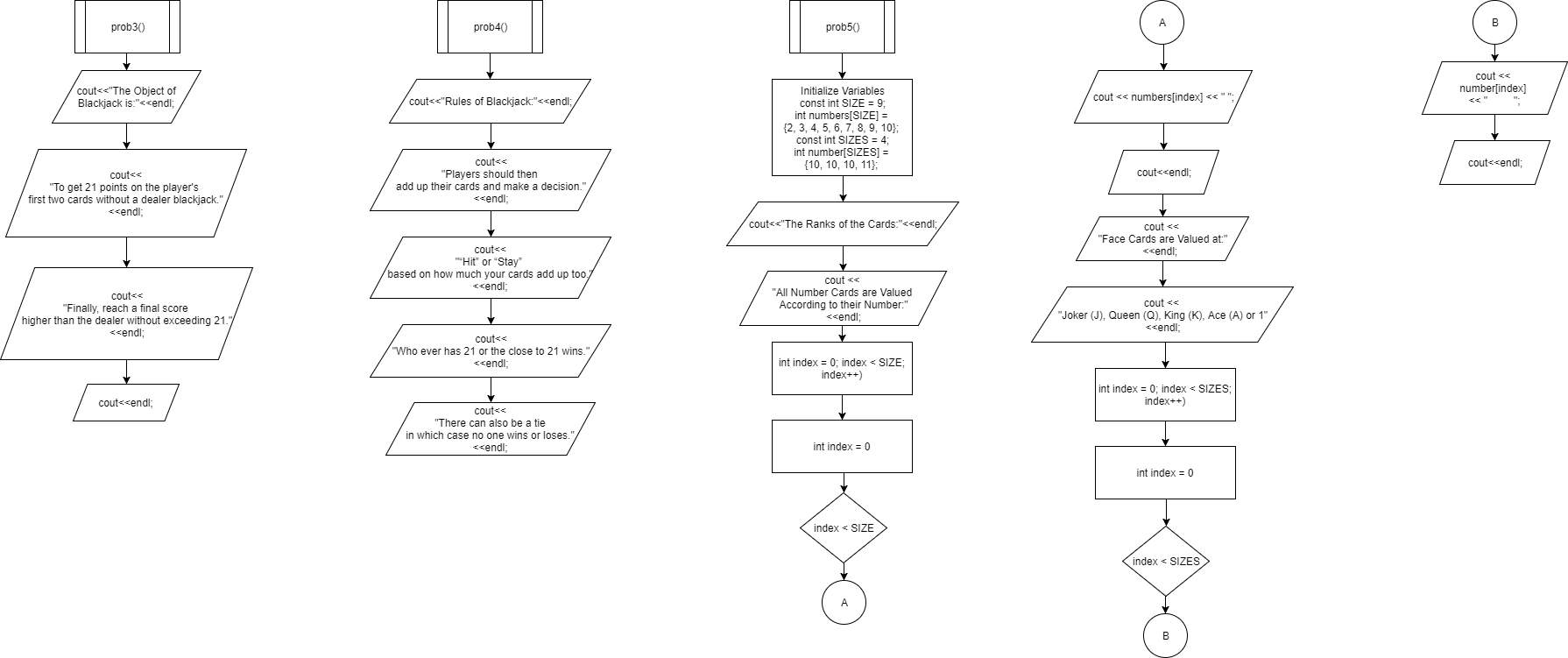
**Flowchart**

Since my flowchart is quite long as one might say, I have broke my flowchart up into smaller pieces making it easier to follow along. To view my complete flowchart, please visit:

<https://github.com/DrakeGerger/Fall_17_Project_2/tree/master/Project_2_Flowcharts>







Pseudo Code

/\*

File: main.cpp

Author: Drake Gerger

Created on December 12, 2017, 5:00 PM

Final Version on December 13, 2017, 8:00 AM

Purpose: Project 2: Blackjack Game

\*/

// Main -> Code Begins Here!

//Set Random Number Seed

// Loop once for each hand

// randomize the cards; set them all to undealt

// Deal the hands. Get two cards for each

// Signal a new hand.

// Get Player's hits. Calculate the score and redisplay after each hit.

// Print the dealt cards, but only the House's second card.

// Ask the Player whether he wants a hit or to stay

// Get the Player's current score to update and check for bust.

// Once the player is done taking hits, check whether he busted

// The Player busted. The House wins.

// If the player didn't bust, then the house takes hits below 17

// The House busted. Player wins

// Compare scores and determine the winner

// Tie. This is called a "push."

// The Player wins

// The House wins

**Constructs & Concepts Utilized**

**iostream Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| cout | 69 | Output Data | Throughout |
| cin | 3 | Input Data | Line  46, 69, 132 |
| void | 14 | Calling Data | Line  20-30 and 33 |
| array | 2 | Memory | Line  231 and 242 |

**cstdlib Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| srand() | 1 | Random # seed | Line 89 |
| rand() | 1 | Generates rand # | Line 306 |

**ctime Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| time | 2 | Set current time | Line 87 and 88 |

**Data Types:**

|  |  |  |
| --- | --- | --- |
| Data Types | Frequency | Location |
| int | 50 | Throughout |
| const int | 15 | Thoughtout |
| char | 1 | 116 |

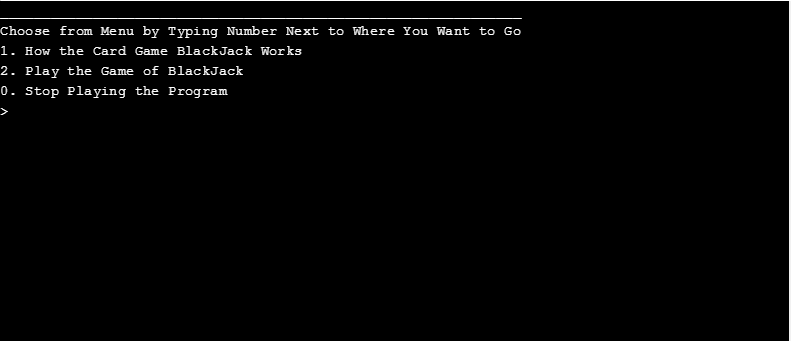
**Conditional Statements:**

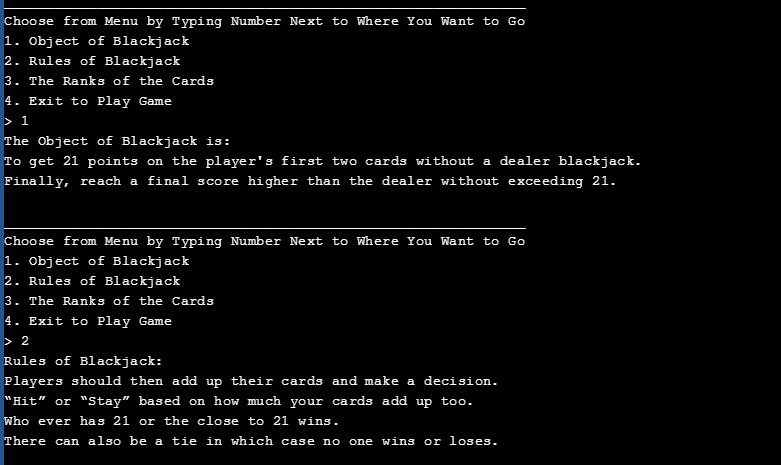
|  |  |  |
| --- | --- | --- |
| Conditional Statement | Frequency | Location |
| if | 8 | Line 135, 152. 166, 173, 266, 281, 307, 320 |
| if/else | 7 | Line 141, 157, 171, 181, 276, 287, 325 |
| if/else if | 9 | Line 138, 177, 268, 270, 272, 274, 283, 285, 323 |
| switch | 2 | Line 48 and 72 |

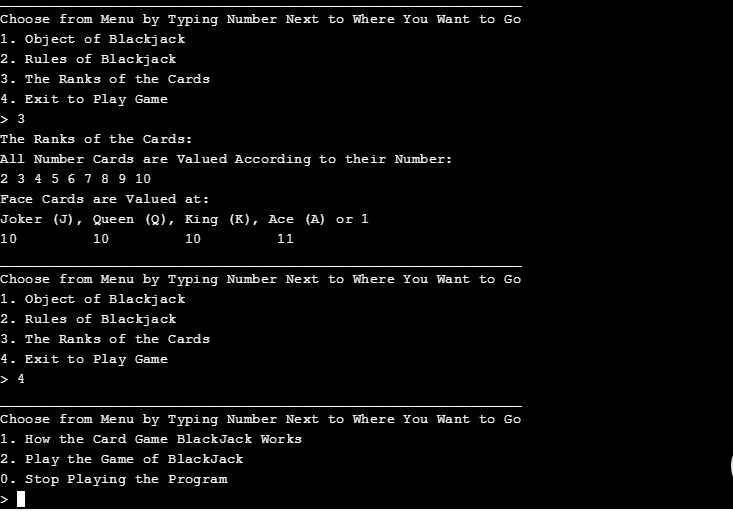
**Loops:**

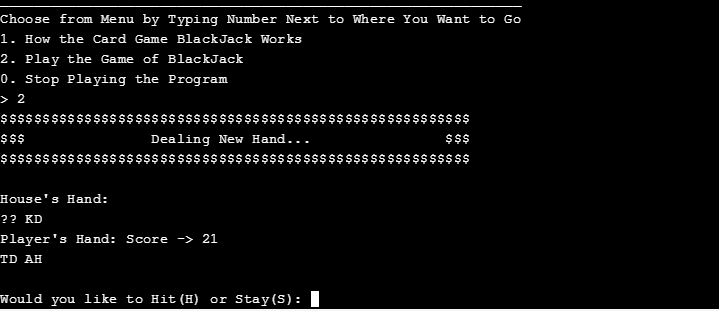
|  |  |  |
| --- | --- | --- |
| Loops | Frequency | Location |
| for | 5 | Line 235, 245, 257, 294, 317 |
| while | 4 | Line 99, 149, 160, 329 |
| do-while | 4 | Line 42, 65, 120, 305 |

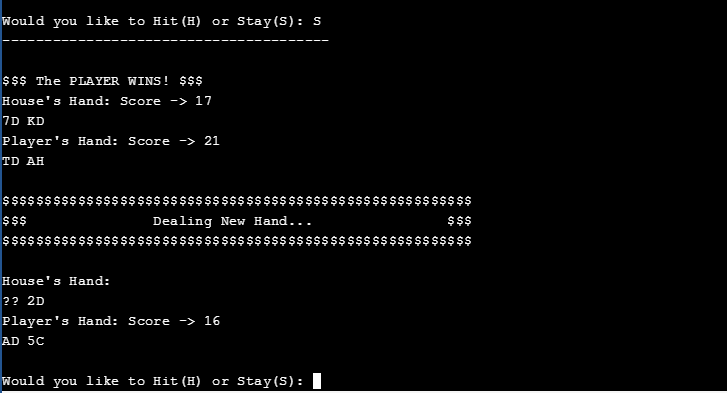
**Proof of a Working Product**

In the event, that my program does not work once it reaches Dr. Lehr, I have provided some screenshots that prove the program did in fact work at the time I turned it in.









**References**

1. Dr. Lehr’s Lectures & Lab

2. “Starting Out with C++: From Control Structures through Objects” Gaddis, Tony. 8th Edition. (Textbook)

3. [www.cplusplus.com](http://www.cplusplus.com) (for the use of 1 and 2 dimensional arrays and search & sort)

**Program**

/\*

\* File: main.cpp

\* Author: Drake Gerger

\* Created on December 11th, 2017, 8:00 PM

\* Purpose: Project 2

\*/

//System Libraries

#include <iostream> //Input/Output Stream Library

#include <ctime>

#include <cstdlib>

using namespace std; //Standard Name-space under which System Libraries reside

//User Libraries

//Global Constants - Not variables only Math/Science/Conversion constants

//Function Prototypes

void prob1();

void prob2();

void menu();

void menu2();

void prob3();

void prob4();

void prob5();

void prob6();

void Shuffle(bool CardsDealt[]);

void PrintCard(int Card);

void PrintHand(int Hand[], const int CardCount);

int GetNextCard(bool CardsDealt[]);

int ScoreHand(int Hand[], const int CardCount);

void PrintScoresAndHands(int HouseHand[],

const int HouseCardCount, int PlayerHand[], const int PlayerCardCount);

//Execution Begins Here!

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

//Declare Variables

int choice;

//Loop the Menu and Problems

do{

//Input Data/Variables

menu();

cout<<"> ";

cin>>choice;

//Process or map the inputs to the outputs

switch(choice){

case 1:prob1();break;

case 2:prob2();break;

default:{

cout<<"Thank You for Playing!"<<endl;

}

}

}while(choice>0&&choice<3);

//Exit the program

return 0;

}

void prob1(){

//Declare Variables

int choice;

//Loop the Menu and Problems

do{

//Input Data/Variables

menu2();

cout<<"> ";

cin>>choice;

//Process or map the inputs to the outputs

switch(choice){

case 1:prob3();break;

case 2:prob4();break;

case 3:prob5();break;

case 4:prob6();break;

default:{

cout<<"Exiting to Main Menu!"<<endl;

}

}

}while(choice>0&&choice<4);

}

void prob2(){

//Declare Variables

time\_t cTime;

time(&cTime);

srand(cTime);

int choice;

bool CardsDealt[52];

int HouseCardCount = 0;

int HouseHand[12];

int PlayerCardCount = 0;

int PlayerHand[12];

// Loop once for each hand

while (true) {

// randomize the cards; set them all to undealt

Shuffle(CardsDealt);

// Deal the hands. Get two cards for each

PlayerHand[0] = GetNextCard(CardsDealt);

HouseHand[0] = GetNextCard(CardsDealt);

PlayerHand[1] = GetNextCard(CardsDealt);

HouseHand[1] = GetNextCard(CardsDealt);

HouseCardCount = 2;

PlayerCardCount = 2;

// Signal a new hand.

cout << "$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$" << endl;

cout << "$$$ Dealing New Hand... $$$" << endl;

cout << "$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$" << endl;

cout << endl;

char PlayerChoice;

bool PlayerHits = true;

int PlayerScore = ScoreHand(PlayerHand, PlayerCardCount);

// Get Player's hits. Calculate the score and redisplay after each hit.

do {

// Print the dealt cards, but only the House's second card.

cout << "House's Hand:" << endl;

cout << "?? ";

PrintCard(HouseHand[1]);

cout << endl;

cout << "Player's Hand: Score -> " << ScoreHand(PlayerHand, PlayerCardCount) << endl;

PrintHand(PlayerHand, PlayerCardCount);

cout << endl;

// Ask the Player whether he wants a hit or to stay

cout << "Would you like to Hit(H) or Stay(S): ";

cin >> PlayerChoice;

cout <<"---------------------------------------"<<endl;

if (PlayerChoice == 'H') {

PlayerHand[PlayerCardCount] = GetNextCard(CardsDealt);

++PlayerCardCount;

} else if (PlayerChoice == 'S') {

PlayerHits = false;

} else {

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

cout << "\*\*\* PLEASE ENTER Hit(H) OR Stay(S)! \*\*\*" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

}

cout << endl;

// Get the Player's current score to update and check for bust.

PlayerScore = ScoreHand(PlayerHand, PlayerCardCount);

} while (PlayerHits && PlayerScore < 22);

// Once the player is done taking hits, check whether he busted

if (PlayerScore > 21) {

// The Player busted. The House wins.

cout << "$$$ The HOUSE WINS! $$$" << endl;

cout << endl;

PrintScoresAndHands(HouseHand, HouseCardCount, PlayerHand, PlayerCardCount);

} else {

// If the player didn't bust, then the house takes hits below 17

int HouseScore = ScoreHand(HouseHand, HouseCardCount);

while (HouseScore < 17) {

HouseHand[HouseCardCount] = GetNextCard(CardsDealt);

++HouseCardCount;

HouseScore = ScoreHand(HouseHand, HouseCardCount);

}

bool HouseBusts = (HouseScore > 21);

if (HouseBusts) {

// The House busted. Player wins

cout << "$$$ The PLAYER WINS! $$$" << endl;

cout << endl;

PrintScoresAndHands(HouseHand, HouseCardCount, PlayerHand, PlayerCardCount);

} else {

// Compare scores and determine the winner

if (PlayerScore == HouseScore) {

// Tie. This is called a "push."

cout << "$$$ Push! $$$" << endl;

PrintScoresAndHands(HouseHand, HouseCardCount, PlayerHand, PlayerCardCount);

} else if (PlayerScore > HouseScore) {

// The Player wins

cout << "$$$ The PLAYER WINS! $$$" << endl;

PrintScoresAndHands(HouseHand, HouseCardCount, PlayerHand, PlayerCardCount);

} else {

// The House wins

cout << "$$$ The HOUSE WINS! $$$" << endl;

PrintScoresAndHands(HouseHand, HouseCardCount, PlayerHand, PlayerCardCount);

}

}

}

}

}

//------------------------------------------------------------------------------

//Function Calling

void menu(){

//Input Data/Variables

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"Choose from Menu by Typing Number Next to Where You Want to Go"<<endl;

cout<<"1. How the Card Game BlackJack Works"<<endl;

cout<<"2. Play the Game of BlackJack"<<endl;

cout<<"0. Stop Playing the Program"<<endl;

}

void menu2(){

//Input Data/Variables

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"Choose from Menu by Typing Number Next to Where You Want to Go"<<endl;

cout<<"1. Object of Blackjack"<<endl;

cout<<"2. Rules of Blackjack"<<endl;

cout<<"3. The Ranks of the Cards"<<endl;

cout<<"4. Exit to Play Game"<<endl;

}

void prob3(){

cout<<"The Object of Blackjack is:"<<endl;

cout<<"To get 21 points on the player's first two cards without a dealer blackjack."<<endl;

cout<<"Finally, reach a final score higher than the dealer without exceeding 21."<<endl;

cout<<endl;

}

void prob4(){

cout<<"Rules of Blackjack:"<<endl;

cout<<"Players should then add up their cards and make a decision."<<endl;

cout<<"“Hit” or “Stay” based on how much your cards add up too."<<endl;

cout<<"Who ever has 21 or the close to 21 wins."<<endl;

cout<<"There can also be a tie in which case no one wins or loses."<<endl;

}

void prob5(){

const int SIZE = 9;

int numbers[SIZE] = {2, 3, 4, 5, 6, 7, 8, 9, 10};

cout<<"The Ranks of the Cards:"<<endl;

cout << "All Number Cards are Valued According to their Number:"<<endl;

for (int index = 0; index < SIZE; index++)

cout << numbers[index] << " ";

cout << endl;

cout << "Face Cards are Valued at:"<<endl;

const int SIZES = 4;

int number[SIZES] = {10, 10, 10, 11};

cout << "Joker (J), Queen (Q), King (K), Ace (A) or 1"<<endl;

for (int index = 0; index < SIZES; index++)

{

cout << number[index] << " ";

}

cout<<endl;

}

void prob6(){

}

void Shuffle(bool CardsDealt[]) {

for (int Index = 0; Index < 52; ++Index) {

CardsDealt[Index] = false;

}

}

void PrintCard(int Card) {

using namespace std;

// Print Rank

const int Rank = (Card % 13);

if (Rank == 0) {

cout << 'A';

} else if (Rank < 9) {

cout << (Rank + 1);

} else if (Rank == 9) {

cout << 'T';

} else if (Rank == 10) {

cout << 'J';

} else if (Rank == 11) {

cout << 'Q';

} else {

cout << 'K';

}

// Print Suit

const int Suit = (Card/13);

if (Suit == 0) {

cout << 'C';

} else if (Suit == 1) {

cout << 'D';

} else if (Suit == 2) {

cout << 'H';

} else {

cout << 'S';

}

}

void PrintHand(int Hand[], const int CardCount) {

using namespace std;

for (int CardIndex = 0; CardIndex < CardCount; ++CardIndex) {

const int NextCard = Hand[CardIndex];

PrintCard(NextCard);

cout << " ";

}

cout << endl;

}

int GetNextCard(bool CardsDealt[]) {

bool CardDealt = true;

int NewCard = -1;

do {

NewCard = (rand() % 52);

if (!CardsDealt[NewCard]) {

CardDealt = false;

}

} while (CardDealt);

return NewCard;

}

int ScoreHand(int Hand[], const int CardCount) {

int AceCount = 0;

int Score = 0;

for (int CardIndex = 0; CardIndex < CardCount; ++CardIndex) {

const int NextCard = Hand[CardIndex];

const int Rank = (NextCard % 13);

if (Rank == 0) {

++AceCount;

++Score;

} else if (Rank < 9) {

Score = Score + (Rank + 1);

} else {

Score = Score + 10;

}

}

while (AceCount > 0 && Score < 12) {

--AceCount;

Score = Score + 10;

}

return Score;

}

void PrintScoresAndHands(int HouseHand[], const int HouseCardCount, int PlayerHand[], const int PlayerCardCount) {

using namespace std;

cout << "House's Hand: Score -> " << ScoreHand(HouseHand, HouseCardCount) << endl;

PrintHand(HouseHand, HouseCardCount);

cout << "Player's Hand: Score -> " << ScoreHand(PlayerHand, PlayerCardCount) << endl;

PrintHand(PlayerHand, PlayerCardCount);

cout << endl;

}