

Project 2
< Blackjack >

CIS 5 - Section 47165
SPRING 2023

By:
Aurelia Juan Sindhunirmala
06/03/2023

Table of Contents

Table of Contents.....	1
Introduction.....	2
Summary.....	2
How the game works.....	2
My Approach to the Game.....	3
Contracts and Concepts Utilized.....	3
Development Summary.....	3
Version 1.....	3
Version 2.....	3
Version 3.....	3
Version 4.....	4
Version 5.....	4
Version 6.....	4
Version 7.....	4
Version 9.....	4
Sample Input and Outputs.....	5
Flowchart.....	6
Reference.....	7
Github Link.....	7
Pseudocode.....	7
Code.....	15

Introduction

Welcome to the captivating world of virtual blackjack! Immerse yourself in the excitement of this computerized version of the classic card game, where you can experience a little bit of fun on your own screen.

Just like in real-life blackjack, the objective remains the same: aim to obtain a hand value of 21 or get as close to it as possible, without surpassing it. In this simplified version, this blackjack is developed using the C++ programming language with the knowledge and materials learned in class.

Are you ready to challenge the digital dealer and rate this blackjack experience out of 10? Let the games begin!

Summary

Project size: 450 lines

Step into the virtual world of blackjack, where the objective is to reach 21 without going over. This simplified version, created using C++, allows you to play against a computerized dealer. Experience the game and rate it out of 10 to help improve future versions. Get ready for an immersive adventure in the virtual "casino"!

How the game works

This is the guideline to play the game blackjack, I created. These rules will be provided in the game as well. The rules are:

1. The game starts with each player placing their bet on the table (max bet: \$1000).
2. The dealer deals two cards to each player, including themselves. The dealer's first card is dealt face up (shown), while the second card is dealt face down (not shown).
3. The value of each card is as follows:
 - a. All numbered cards are worth their face value
 - b. Face cards (such as Jacks, Queens, and Kings) are worth 10, and
 - c. Aces are worth 1 or 11 based on the system itself.
4. Players can choose to hit (receive another card) or stand (keep their current card) in order to get as close to 21 as possible without going over. Players can continue to hit until they decide to stand, or until they go over 21, or known as a bust.
5. Once the player has completed their turn, the dealer reveals their hole card and hits or stands according to a set of predetermined rules"

6. If the dealer busts, the player wins their bets. If the dealer does not bust, then the player and the dealer with the hand value closest to 21 without going over win. If the player's hand value is the same as the dealer's, it is a tie, and the player's bet is returned.

My Approach to the Game

The game will start with an opening screen that consists of a welcome sign and a menu to choose from. With every menu chosen by the user and accomplishing each menu, the system will ask if the user with to proceed or repeat the game or exit the game instead. Ratings will only appear if the user finishes a game.

Contracts and Concepts Utilized

This project cover all the materials learned in the Spring 2023 semester. The chapters include Introduction to C++, Expressions and Interactivity, Making Decisions, Loops and Files, Functions, Arrays, and Searching and Sorting Arrays. For detailed implementation in the code, please refer to the spreadsheet attached to this file. In the spreadsheet, there will be two pages which are for Project 2 and 1 itself.

Development Summary

Version 1

The first version of this project 2, converts the last version of Project 1 to a version that applies functions. Repeated codes were formed in one function and re-called to be used again. Some functions were for the menu, the first option of the table of contents, the second, third, and rating.

Version 2

In the second version, I revised a couple of mistakes that were found in this code. I also add an additional function to exit the program in each part of the game. With the exit function, it will allow the program to ask the user if the user wants to stay in the game or not.

Version 3

In the third version, I added the face cards and card suits for the game. As for Project 1, this blackjack has only been a game to obtain a number that is 21 or closer to 21. Now, for the second project, I added the face cards and card suits that we have in a card game.

Version 4

In the fourth version, the modification that I did was simplifying the program's code and adding a returning boolean and defaulted arguments.

Version 5

This version is where arrays and sorts are implemented in the program. The program will run but no specific changes were made for the output.

Version 6

I fixed and try a few things in the code. No specific things were made and influenced the output.

Version 7

In this version I tried to implement overloading functions and fix some problems that occurred in the output.

Version 8

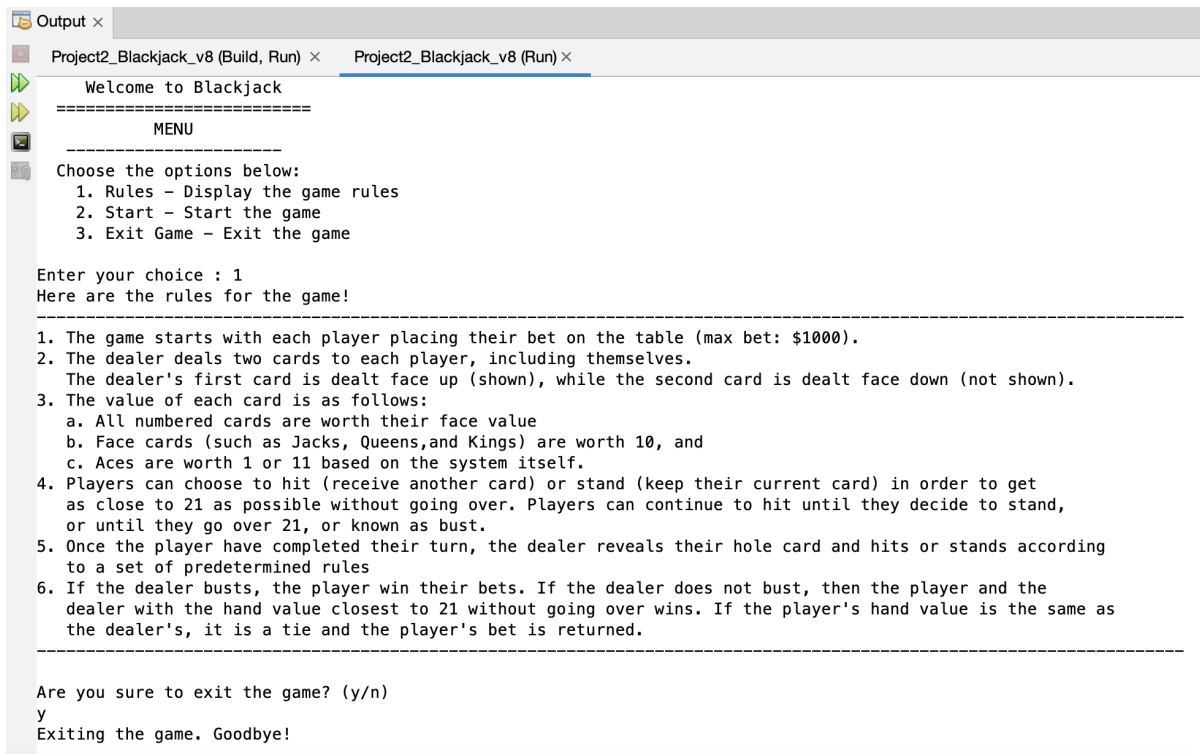
Final version with couple of mistakes that were fixed on the next version.

Version 9

This version is the last version of the second project. Users can run the code and play the game. Starting off with the menu, every time user inputs an invalid option, the code will run again and ask for another input. After finishing a round of the game, players are asked to rate the game. The different between this version and version 8 are correction of little mistakes and cleaning up the codes.

Sample Input and Outputs

This is how the program will output with the latest version:



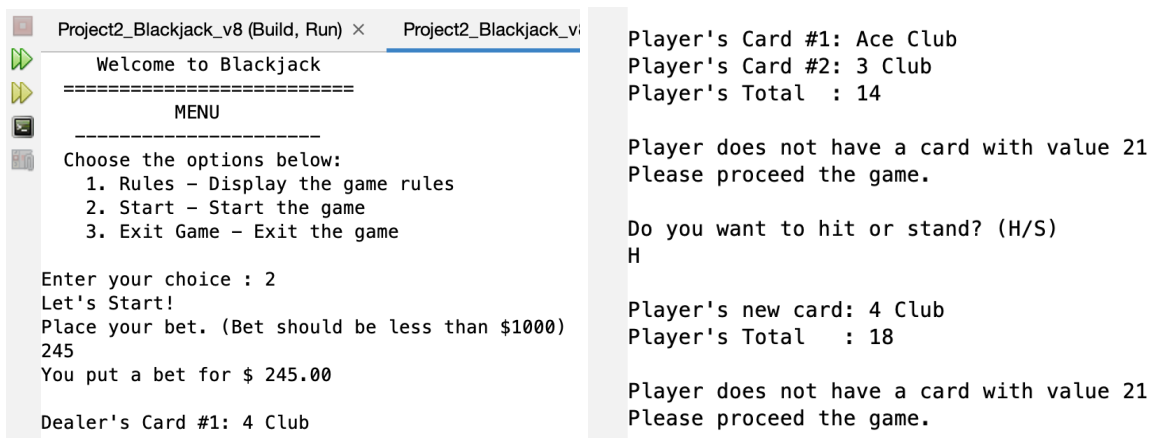
```

Output ×
Project2_Blackjack_v8 (Build, Run) × Project2_Blackjack_v8 (Run) ×
Welcome to Blackjack
=====
MENU
-----
Choose the options below:
1. Rules - Display the game rules
2. Start - Start the game
3. Exit Game - Exit the game

Enter your choice : 1
Here are the rules for the game!
-----
1. The game starts with each player placing their bet on the table (max bet: $1000).
2. The dealer deals two cards to each player, including themselves.
   The dealer's first card is dealt face up (shown), while the second card is dealt face down (not shown).
3. The value of each card is as follows:
   a. All numbered cards are worth their face value
   b. Face cards (such as Jacks, Queens, and Kings) are worth 10, and
   c. Aces are worth 1 or 11 based on the system itself.
4. Players can choose to hit (receive another card) or stand (keep their current card) in order to get
   as close to 21 as possible without going over. Players can continue to hit until they decide to stand,
   or until they go over 21, or known as bust.
5. Once the player have completed their turn, the dealer reveals their hole card and hits or stands according
   to a set of predetermined rules
6. If the dealer busts, the player win their bets. If the dealer does not bust, then the player and the
   dealer with the hand value closest to 21 without going over wins. If the player's hand value is the same as
   the dealer's, it is a tie and the player's bet is returned.
-----

Are you sure to exit the game? (y/n)
y
Exiting the game. Goodbye!
  
```

The picture above shows the output when user chose 1 from the menu. It will print out the rules of the game.



<pre> Project2_Blackjack_v8 (Build, Run) × Project2_Blackjack_v8 (Run) × Welcome to Blackjack ===== MENU ----- Choose the options below: 1. Rules - Display the game rules 2. Start - Start the game 3. Exit Game - Exit the game Enter your choice : 2 Let's Start! Place your bet. (Bet should be less than \$1000) 245 You put a bet for \$ 245.00 Dealer's Card #1: 4 Club </pre>	<pre> Player's Card #1: Ace Club Player's Card #2: 3 Club Player's Total : 14 Player does not have a card with value 21 Please proceed the game. Do you want to hit or stand? (H/S) H Player's new card: 4 Club Player's Total : 18 Player does not have a card with value 21 Please proceed the game. </pre>
---	---

The picture above is when user start playing the game.

```

Do you want to hit or stand? (H/S)
H

Player's new card: 9 Club
Player's Total   : 27

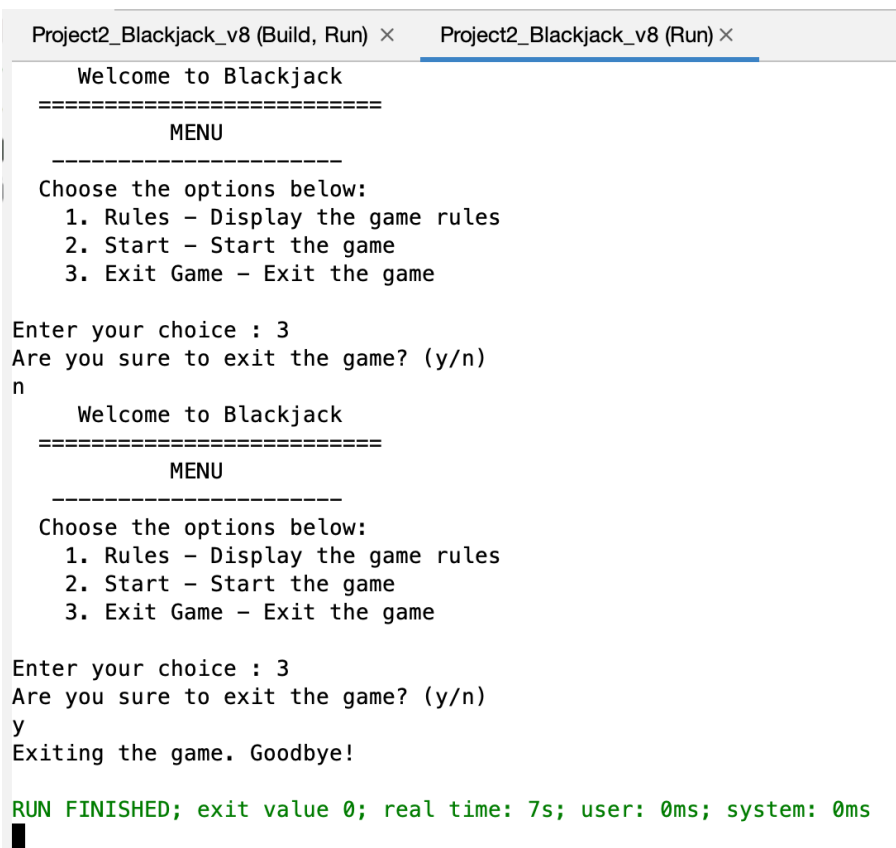
Player busts, dealer wins!
You lose your bet of $245.00

THANK YOU FOR VISITING BLACKJACK!
Please rate us starting from 1-10
Enter a number between 1 and 10: 9
You have rated this program as good.
Thank you for rating this program! Hope you enjoy your visit!

Are you sure to exit the game? (y/n)
y
Exiting the game. Goodbye!

RUN FINISHED; exit value 0; real time: 1m 16s; user: 0ms; system: 0ms

```



```

Project2_Blackjack_v8 (Build, Run) x  Project2_Blackjack_v8 (Run) x
Welcome to Blackjack
=====
MENU
-----
Choose the options below:
  1. Rules - Display the game rules
  2. Start - Start the game
  3. Exit Game - Exit the game

Enter your choice : 3
Are you sure to exit the game? (y/n)
n
Welcome to Blackjack
=====
MENU
-----
Choose the options below:
  1. Rules - Display the game rules
  2. Start - Start the game
  3. Exit Game - Exit the game

Enter your choice : 3
Are you sure to exit the game? (y/n)
y
Exiting the game. Goodbye!

RUN FINISHED; exit value 0; real time: 7s; user: 0ms; system: 0ms

```

This is what it shows when user pick number 3 from the menu.

Flowchart

There is a flowchart file attached for this project.

Reference

1. Dr. Lehr's Lectures and Lab
2. Gaddis, T. (2017). Starting out with C++ from control structures to objects (9th ed.). Pearson.

Github Link

The project are posted in the github with the link down below:

<https://github.com/AureliaSindhu/CIS5-Final-Project>

Pseudocode

Include all system libraries: iostream, iomanip, cstdlib, ctime, fstream, string, cmath, vector

Function prototypes: first, menu(), display, def(int), toc1(), toc2(), toc3(), Rate(), vldRate(), exitGm(), printCS(int), printFV(int), slSt(int [], int), linSc(int [], int, int)

Declaring global constants

```
const int OPTIONS = 3
```

```
const string menuOpt[OPTIONS][2]
```

Begin main function with parameters int argc, char** argv

Set random number seed using srand function with time(0) as argument

Begin main function

```
DO
```

```
    first()
```

```
    WHILE not exitGm()
```

End function

Begin Function first

```
    Declare Integer choice, inp
```

```
    Static Integer gmCnt = 0
```

```
    output "Welcome to Blackjack"
```

```
    output "=====
```

```
    Do
```

```
        Call Function Menu, passing choice
```



```

    Switch choice
        Case 1: Call Function toc1
        Case 2: Call Function toc2
        Case 3: Call Function toc3
        Default: Call Function def, passing inp
    End Switch
While inp < 3

    Increment gmCnt
    output "Number of games played: " + gmCnt
End Function

Begin Function Menu, passing option
    Declare Boolean vldInp = False
    option = 0
    WHILE not vldInp
        Call Function display
        output ""
        output "Enter your choice" + option + ": "
        input option
        If option < 1 or option > 3
            Output Input invalid.
        ELSE vldInp is true
    End Function

Begin Function Display
    Declaring
        const int OPTIONS equals to 3
        const string menuOpt[OPTIONS][2]
            {"1. Rules", "Display the game rules"},
            {"2. Start", "Start the game"},
            {"3. Exit Game", "Exit the game"}
    output "MENU"
    output "-----"
    output "Choose the options below:"
    Generate a for loop to print out the menu that is sorted in
    2D arrays

```

End Function

Begin Function Def

 output typing a value will exit the program.

End Function

Begin Function TOC1

 output the rules of the game

End Function

Begin Function TOC2

 Declaring variable

 int : card, faceVal, sum, sum2

 char : hsAns

 float : pBet

 Output start game

 Output betting time

 Input pBet

 IF pBet is greater than 1000

 Output invalid value

 ELSE pBet is within range

 IF pBet is less than 0

 Set pBet to the absolute value of pBet

 Output value of the user's bet in format

 Set sum to 0

 Generate a for loop which shuffles the deck for the dealer

 Output dealer's first card

 Declare

 int : pCrd[2], numPC = 2

 Generate a for loop which shuffles the deck for the player

 Generate a bubble sort for the player's cards

 Call the slSt function

 Output the player's first and second card

 Set sum2 to 0

 Generate a for loop to calculate player's total card

Output the value of the player's total card

```

WHILE sum and sum2 is less than 21, sum is less than sum 2,
sum2 is less than sum, and sum and sum2 is not equal to 21
    Generate a linear search if player's total card is
    not 21
        Declare scVal to 21
        Create a bool statement of found which calls
        linSc function
        IF bool is found
            output "Player has a card value"
        ELSE
            output "Player does not have a card with
            value 21"
            output "Please proceed the game"

```

Output message to ask if the player wants to hit or stay

Input hsAns

```

IF hsAns is 'H' or 'h'
    Generate a for loop to add cards for player
    Output new card
    Output player's new total

    IF sum2 is greater than 21
        Output message player busts and dealer wins!
        Output the total loss of player's bet
        Call rate function
        Call exitGm function

    IF sum2 is equal to 21
        Output message player wins!!
        Output the player's bet
        Call rate function
        Call exitGm function

```

```

ELSE IF hsAns is 'S' or 's'
    Generate a for loop for dealers second card
    Output dealer's second card
    Output dealer's total

    IF sum is less than 17
        Initialize random card
        Output dealer's new card
        Call the printFV function
        Call the printCS function
        output dealer's total

        IF sum is greater than 21
            output dealer busts, player wins!
            output congrats, player won their bet
            Call the rate function
            Call the exit function
    IF sum is less than or equal to 21 and sum2 is
    less than or equal to 21
        IF sum is greater than sum2
            output dealer wins!
            output player's lost their bet
            Call the rate function
            Call the exit function

    ELSE IF sum is greater than 21
        output dealer busts, player wins!
        output congrats, player won their bet
        Call the rate function
        Call the exit function

    ELSE
        output it's a tie!
        output player got their bet back
        Call the rate function
        Call the exit function

```

End Function

Begin Function linSc

 Generate a for loop to search a particular number (21)

 IF arr[i] is key

 return true;

 return false

End Function

Begin Function printCS

 switch sign

 CASE 1

 output "diamond"

 Break

 CASE 1

 output "club"

 Break

 CASE 1

 output "heart"

 Break

 CASE 1

 output "spade"

 Break

End Function

Begin Function printFV

 IF value is 1 or 11

 output "Ace "

 ELSE IF value is greater or equal than 2 or less than 10

 output value

 ELSE IF value is equal to 10

 output "Jack "

 ELSE IF value is equal to 11

 output "Queen "

 ELSE IF value is equal to 12

 output "King "

End Function

Begin Function slSt

 Generate a single dimensioned as function arguments

 Generate a map to found the minimum val

End Function

Begin Function toc 3

 Call exitGm function

End Function

Begin Function Rate

 output "THANK YOU FOR VISITING BLACKJACK!"

 output "Please rate us starting from 1-10"

 Set inpRate to vldRate function

 output "You have rated this program as " ratings[inpRate - 1]

 output "Thank you for rating this program! Hope you enjoy your visit!"

 Return true

End Function

Begin Function Rate

 Input inpRate

 Declare variable

 string: rate

 look through the file which line is the input located

 Return rate(ratings)

End Function

Begin Function vldRate

 Declare variable

 int : inpRate;

 bool : validRating = false

 WHILE it is not validRating

 output "Enter a number between 1 and " ratings.size

 input inpRate

 IF inpRate is greater than or equal to 1 and less than or ratings.size

```
        validRating is true
    ELSE
        output "Invalid rating. Please try again!"
    Return inpRate
End Function

Begin Function exitGm
    Declare variable
        char: ext
    output "are you sure to exit the game? (y/n)"
    Input ext
    IF ext is 'Y' or 'y'
        output "Exiting the game. Goodbye!"
        Call exit function
    ELSE
        Call first function
    Return true
End Function
```

Code

```

/*
 * File:    main.cpp
 * Author: Aurelia Sindhunirmala
 *
 * Created on June 7, 2023, 11:30 PM
 * Purpose: Version 9 of Project 2
 *
 */

// System Libraries
#include <iostream>      //Input/Output library
#include <iomanip>        //Format library
#include <ctime>          //Set Random library
#include <cstdlib>        //Random Functions
#include <fstream>        //File Stream Library
#include <string>         //String
#include <cmath>          //Math Library
#include <vector>         //Vector Library
using namespace std;

//User Libraries

//Global Constants Math/Physics/Chemistry/Conversions ONLY!!!!
const int OPTIONS = 3;
const string menuOpt[OPTIONS][2];

//Function Prototypes
void first();                //First page of the game
void Menu(int&);             //Menu for the first page
void display();              //Display menu
void def(int);               //Default switch case for menu
void toc1();                 //Menu option 1
bool toc2();                 //Menu option 2
void toc3();                 //Menu option 3
bool Rate();                 //Rate the game
int vldRate(const vector<string>&); //Validating Rate Input
bool exitGm();               //Exit game
void printCS(int);           //Print Card Suites
void printFV(int);           //Print Card Face Val

```



```

void slSt(int [], int);           //Selection sort
bool linSc(int [], int , int);    //Linear search

//Execution Begins HERE!!!!
int main(int argc, char** argv) {
    //Set random number seed
    srand(static_cast <unsigned int>(time(0)));

    //Declare variables

    //Initialize Inputs

    //Map Inputs to Outputs - Process
    do{
        first();
    }
    while (!exitGm());
    //Display output

    //Exit stage right
    return 0;
}

void first(){ //First page of the game
    //Declare variables
    int choice, inp;
    static int gmCnt = 0;

    //Display output
    cout<<setw(25)<<"Welcome to Blackjack"<<endl;
    cout<<setw(28)<<"===== "<<endl;

    do{ //choice of menu
        Menu(choice);
        switch(choice){
            case 1:    toc1();break;    //calling function toc1
            case 2:    toc2();break;    //calling function toc2
            case 3:    toc3();break;    //calling function toc3
            default:    def(inp);
        }
    }
}

```

```

    }
    while (choice < 3);

    // Increment gameCount
    gmCnt++;
    cout << "Number of games played: " << gmCnt << endl;
}

void Menu(int& option){    //Menu for the first page
    bool vldInp = false;
    option = 0;
    while (!vldInp){
        display();
        cout << endl << "Enter your choice : ";
        cin >> option;

        if (option == '1' || option == '2' || option == '3') {
            cout << option;
        }
        else if (option < 1 || option > 3){ //If input is not within the
requirements
            cout << "Input invalid." << endl << endl;
        }
        else{ vldInp = true;}
    }
}

void display(){
    string menuOpt[OPTIONS][2] = {    //2D arrays for printing out the
menu
        {"1. Rules", "Display the game rules"},
        {"2. Start", "Start the game"},
        {"3. Exit Game", "Exit the game"}
    };

    cout << setw(16) << "MENU" << endl;
    cout << setw(25) << "-----" << endl;
    cout << setw(27) << "Choose the options below:" << endl;

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

```

```

        cout << setw(12);
        if (i == OPTIONS - 1){
            cout << setw(16);
        }
        cout << menuOpt[i][0] << " - " << menuOpt[i][1] << endl;
//Calling the menu with 2D arrays
    }
}

void def(int inp = 0){ //Default switch case for menu
    cout<<endl<<"Typing "<<inp<<" exits the program."<<endl;
}

void toc1(){ //Menu option 1: Rules of the Game
    cout << "Here are the rules for the game!" << endl;
    cout << "-----" << endl;
    cout << "-----" << endl;
    cout << "1. The game starts with each player placing their bet on
the table (max bet: $1000)." << endl;
    cout << "2. The dealer deals two cards to each player, including
themselves." << endl;
    cout << "    The dealer's first card is dealt face up (shown),
while the second card is dealt face down (not shown)." << endl;
    cout << "3. The value of each card is as follows: " << endl;
    cout << "    a. All numbered cards are worth their face value" <<
endl;
    cout << "    b. Face cards (such as Jacks, Queens,and Kings) are
worth 10, and" << endl;
    cout << "    c. Aces are worth 1 or 11 based on the system itself."
<< endl;
    cout << "4. Players can choose to hit (receive another card) or
stand (keep their current card) in order to get " << endl;
    cout << "    as close to 21 as possible without going over. Players
can continue to hit until they decide to stand, " << endl;
    cout << "    or until they go over 21, or known as bust."<<endl;
    cout << "5. Once the player have completed their turn, the dealer
reveals their hole card and hits or stands according" << endl;
    cout << "    to a set of predetermined rules" << endl;
}

```

```

    cout << "6. If the dealer busts, the player win their bets. If the
dealer does not bust, then the player and the" << endl;
    cout << "    dealer with the hand value closest to 21 without going
over wins. If the player's hand value is the same as " << endl;
    cout << "    the dealer's, it is a tie and the player's bet is
returned."<<endl;

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

    cout << endl;
    exitGm();
}

bool toc2(){    //Menu option 2
    //Declaring variables
    int card, faceVal, sum, sum2;
    char hsAns;
    float pBet;

    cout << "Let's Start!" << endl;
    cout << "Place your bet. (Bet should be less than $1000)"<<endl;
    cin >> pBet;

    if (pBet > 1000){
        cout << "Value exceed the maximum rule." << endl << endl;
    }
    else if (pBet < 0){
        pBet = abs(pBet);
    }
    cout << fixed << showpoint << setprecision(2);
    cout << "You put a bet for $ " << pBet << endl << endl;

    //Shuffle deck for dealer
    sum = 0;
    for (int i = 0; i < 1; i++) {
        card = rand() % 52 + 1;           //[1,52]
        faceVal = (card % 11) + 1;       //[1,11]
        sum += faceVal;
        cout << "Dealer's Card #1" << ": ";

```

```

                printFV(faceVal);                                // Printing Cards Face
Value
                printCS(rand() % 4 + 1);                        // Randomly generate and
print card sign
                cout << endl << endl;
            }

//Shuffle deck for player
int pCrd[2];
int numPC = 2;
for (int i = 0; i < numPC; i++) {
    card = rand() % 52 + 1;    //[1,52]
    faceVal = (card % 11) + 1;    //[1,11]
    pCrd[i] = faceVal;
    cout << "Player's Card #" << i + 1 << ": ";
    printFV(faceVal);
    printCS(rand() % 4 + 1); // Randomly generate and print card
sign
    cout << endl;
}

// Bubble sort the player's cards
for (int i = 0; i < numPC - 1; i++) {
    for (int j = 0; j < numPC - i - 1; j++) {
        if (pCrd[j] > pCrd[j + 1]) {
            int temp = pCrd[j];
            pCrd[j] = pCrd[j + 1];
            pCrd[j + 1] = temp;
        }
    }
}

// Selection sort the player's cards
slSt(pCrd, numPC);

// Calculate the sum of player's cards
sum2 = 0;
for (int i = 0; i < numPC; i++) {
    sum2 += pCrd[i];
}

```

```

// Display sum to player
cout << "Player's Total  : " << sum2 << endl << endl;

// Additional cards for player
while (sum <= 21 || sum2 <= 21 || sum < sum2 || sum2 < sum
      || sum != 21 || sum2 != 21) {

    // Linear search for a specific card value
    int scVal = 21;
    bool found = linSc(pCrd, numPC, scVal);
    if (found) {
        cout << endl << "Player has a card with value " << scVal
<< endl;
    } else {
        cout << "Player does not have a card with value " << scVal
<< endl;
        cout << "Please proceed the game." << endl << endl;
    }

    cout << "Do you want to hit or stand? (H/S)" << endl;
    cin >> hsAns;
    cout << endl;

    // If player choose to hit
    if (hsAns == 'H' || hsAns == 'h') {
        card = rand() % 52 + 1;      //[1,52]
        faceVal = (card % 11) + 1;    //[1,11]
        sum2 += faceVal;
        cout << "Player's new card: ";
        printFV(faceVal);
        printCS(rand() % 4 + 1); // Randomly generate and print
card sign
        cout << endl;
        cout << "Player's Total  : " << sum2 << endl << endl;

        // If player's card sum exceeds 21, player busts and loses
the game
        if (sum2 > 21) {
            cout << "Player busts, dealer wins!" << endl;

```

```

        cout << "You lose your bet of " << "$" << pBet <<
endl;

        cout << endl;

        Rate();
        exitGm();
    }

    // If player's card sum is exactly 21, player wins the
game
    else if (sum2 == 21) {
        cout << "Player wins!" << endl;
        cout << "Congrats! You win your bet of " << "$" <<
pBet << endl;
        cout << endl;

        Rate();
        exitGm();
    }
}
// If player choose to stand, dealer starts their turn
else if (hsAns == 'S' || hsAns == 's') {
    // Dealer's 2nd card
    card = rand() % 52 + 1;    //[1,52]
    faceVal = (card % 11) + 1;    //[1,11]
    sum += faceVal;
    cout << "Dealer's Card #2: ";
    printFV(faceVal);
    printCS(rand() % 4 + 1); // Randomly generate and print
card sign
    cout << endl;
    cout << "Dealer's Total  : " << sum << endl << endl;

    // Dealer's additional cards
    if (sum < 17) {
        card = rand() % 52 + 1;    //[1,52]
        faceVal = (card % 11) + 1;    //[1,11]
        sum += faceVal;
        cout << "Dealer's new card: ";
        printFV(faceVal);
    }
}

```

```

        printCS(rand() % 4 + 1); // Randomly generate and
print card sign
        cout << endl;
        cout << "Dealer's Total    : " << sum << endl << endl;

        // If dealer's card sum exceeds 21, dealer busts and
player wins
        if (sum > 21) {
            cout << "Dealer busts, player wins!" << endl;
            cout << "Congrats! You win your bet of " << "$" <<
pBet << endl;

            cout << endl;

            Rate();
            exitGm();
        }
    }

    // If neither player nor dealer busts, compare their card
sums
    if (sum <= 21 && sum2 <= 21) {
        if (sum > sum2) {
            cout << "Dealer wins!" << endl;
            cout << "You lose your bet of " << "$" << pBet <<
endl;

            cout << endl;

            Rate();
            exitGm();
        }
        else if (sum < sum2) {
            cout << "Player wins!" << endl;
            cout << "Congrats! You win your bet of " << "$" <<
pBet << endl;

            cout << endl;

            Rate();
            exitGm();
        }
        else {

```



```

        cout << "It's a tie!" << endl;
        cout << "You get your bet back!" << endl;
        cout << endl;

        Rate();
        exitGm();
    }
}

}

}

}

bool linSc(int arr[], int size, int key) { //Linear search
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            return true;
        }
    }
    return false;
}

void printCS(int sign) {    //Printing Card Suites
    switch (sign) {
        case 1: cout << "Diamond"; break;
        case 2: cout << "Club"; break;
        case 3: cout << "Heart"; break;
        case 4: cout << "Spade"; break;
        default: break;
    }
}

void printFV(int value) {    //Printing Face Value for cards
    if (value == 1 || value == 11) {
        cout << "Ace ";
    }
    else if (value >= 2 && value < 10) {
        cout << value << " ";
    }
    else if (value == 10) {
        cout << "Jack ";
    }
}

```

```

    }
    else if (value == 11) {
        cout << "Queen ";
    }
    else if (value == 12) {
        cout << "King ";
    }
}

void slSt(int arr[], int size) {    //Selection sort
    for (int i = 0; i < size - 1; i++) {
        int indx = i;
        for (int j = i + 1; j < size; j++) {
            if (arr[j] < arr[indx]) {
                indx = j;
            }
        }
        // Swap the found minimum val
        int temp = arr[indx];
        arr[indx] = arr[i];
        arr[i] = temp;
    }
}

void toc3(){    //Menu option 3
    exitGm();
}

bool Rate(const vector<string>& ratings){    //Rate the game
    cout << "THANK YOU FOR VISITING BLACKJACK!" <<endl;
    cout << "Please rate us starting from 1-10" << endl;

    int inpRate = vldRate(ratings);

    //Printing out the category for the rating
    cout << "You have rated this program as " << ratings[inpRate - 1]
    << "." << endl;
    cout << "Thank you for rating this program! Hope you enjoy your
    visit!" << endl << endl;
}

```

```

        return true;
    }

    bool Rate(){
        //Calling the file and declaring variables inside the file
        fstream in;
        string rate;

        //Reading the file
        in.open("rate.dat", ios::in);
        vector<string> ratings;
        while (getline(in, rate)){
            ratings.push_back(rate);
        }
        in.close();

        return Rate(ratings);
    }

    int vldRate(const vector<string>& ratings) {    //Checking if the input
    is valid or not
        int inpRate;
        bool valid = false;

        //Asking user for input
        while (!valid) {
            cout << "Enter a number between 1 and " << ratings.size() <<
": ";
            cin >> inpRate;

            //Validating user input
            if (inpRate >= 1 && inpRate <= ratings.size()) {
                valid = true;
            } else {
                cout << "Invalid rating. Please try again!" << endl;
            }
        }
        return inpRate;
    }

```

```
bool exitGm(){ //Option to exit game
    char ext;
    cout << "Are you sure to exit the game? (y/n)" << endl;
    cin >> ext;

    //User input Y or N
    (ext == 'Y' || ext == 'y') ?
        cout << "Exiting the game. Goodbye!" << endl,
        exit(0) :
        //If user choose N or n
        first();
    return true;
}
```