Project 1

<Blackjack>

CSC-17A-48668

Name: Christian Fuentes

Date: November 14, 2021

Introduction

Game: Blackjack

Blackjack is a card game where the goal of the game is to get the cards to add up to 21. Players can either stand (don't draw a card) or hit (draw a card) they can hit as many times as the player demands but if their total goes over 21 they bust (lose). The game starts with each player and dealer getting 2 cards but the dealer only reveals one. The player is asked if they want to hit or stand then after all players are done getting their cards the dealer will flip over their second card. If the dealer has a hand whose total is less than 17 then the dealer will draw until their hand is equal to or over 17. Players will compare hands with the dealer and if their hand is over 21 they lose but if it's not they compare who has the biggest hand between them and the dealer.

Summary

Project size: 356 lines (final version)

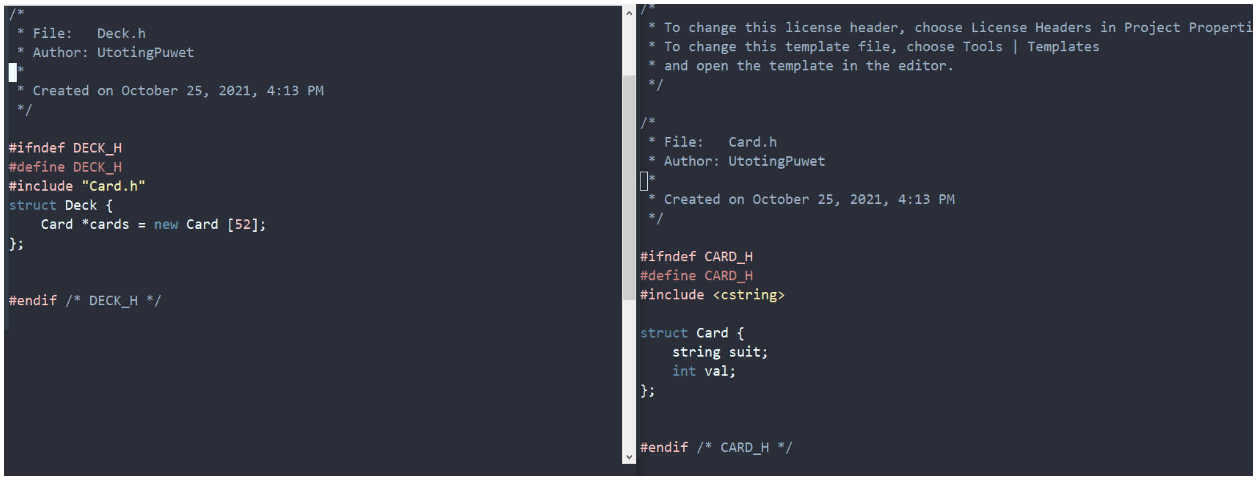
This project is only based on a couple of chapters in Gaddis but it's my biggest version of Blackjack so far. 356 lines in only the main.cpp file and doesn't count the other header files. A lot of lines were taken up from modularizing the game compared to my last project in CIS-5. I think the game is a lot more readable this time as well because the logic is easier to follow.

In my opinion, I have really enjoyed using pointers even if there were some complications along the way I think pointers are way better than using static arrays. I also loved using structures as they made the code way cleaner than having a bunch of jumbled variables.

Development

Creating_Deck_V1

Creating_Deck_V1 is the first version I made when starting this project and its header files contained the structures that would be very important to this project. I used mod functions to create cards.

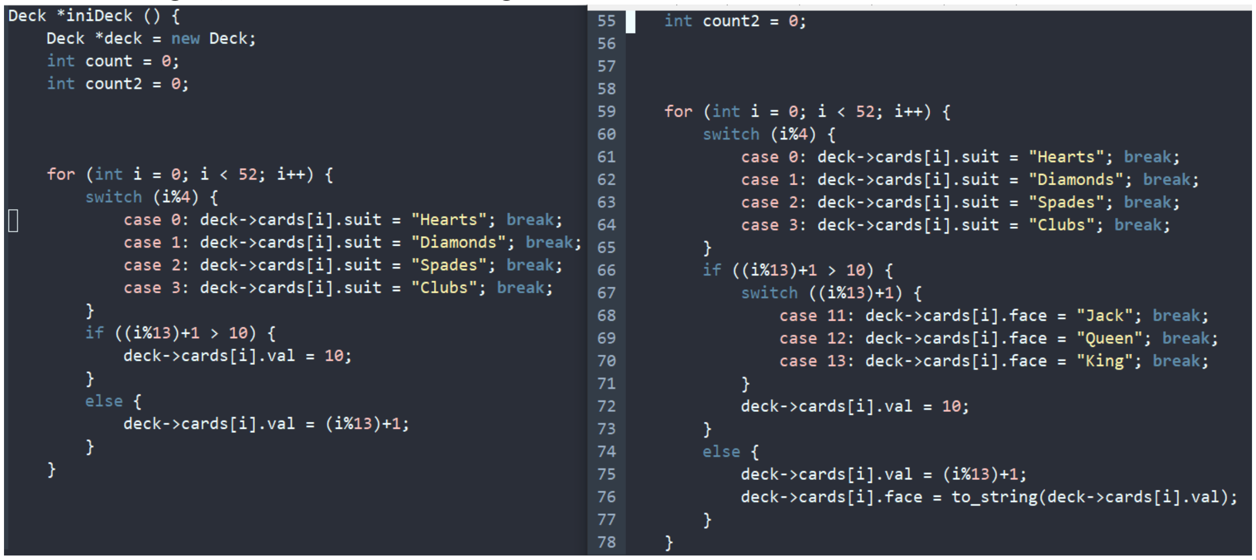


```
/*
 * File:   Deck.h
 * Author: UtotingPuwet
 *
 * Created on October 25, 2021, 4:13 PM
 */

#ifndef DECK_H
#define DECK_H
#include "Card.h"
struct Deck {
    Card *cards = new Card [52];
};


#endif /* DECK_H */
```

```
/*
 * To change this license header, choose License Headers in Project Properti
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   Card.h
 * Author: UtotingPuwet
 *
 * Created on October 25, 2021, 4:13 PM
 */

#ifndef CARD_H
#define CARD_H
#include <cstring>

struct Card {
    string suit;
    int val;
};


#endif /* CARD_H */
```

(Screenshot of header files of Creating_Deck_V1)

Creating_Deck_V2 is almost the same as Creating_Deck_V1 but I added the variable "face" and used the "cstring" function called "to_string" in order to turn all numbers into a string. I had to include this because of the Jack, King, and Queen card that needed to be included. Then I added the Jack, King, Queen, and Ace card logic into the "Deck" structure.



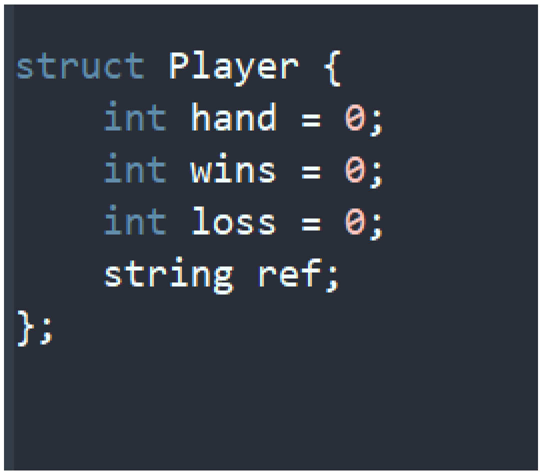
```
Deck *iniDeck () {
    Deck *deck = new Deck;
    int count = 0;
    int count2 = 0;



    for (int i = 0; i < 52; i++) {
        switch (i%4) {
            case 0: deck->cards[i].suit = "Hearts"; break;
            case 1: deck->cards[i].suit = "Diamonds"; break;
            case 2: deck->cards[i].suit = "Spades"; break;
            case 3: deck->cards[i].suit = "Clubs"; break;
        }
        if ((i%13)+1 > 10) {
            deck->cards[i].val = 10;
        }
        else {
            deck->cards[i].val = (i%13)+1;
        }
    }
```

```
    int count2 = 0;



    for (int i = 0; i < 52; i++) {
        switch (i%4) {
            case 0: deck->cards[i].suit = "Hearts"; break;
            case 1: deck->cards[i].suit = "Diamonds"; break;
            case 2: deck->cards[i].suit = "Spades"; break;
            case 3: deck->cards[i].suit = "Clubs"; break;
        }
        if ((i%13)+1 > 10) {
            switch ((i%13)+1) {
                case 11: deck->cards[i].face = "Jack"; break;
                case 12: deck->cards[i].face = "Queen"; break;
                case 13: deck->cards[i].face = "King"; break;
            }
            deck->cards[i].val = 10;
        }
        else {
            deck->cards[i].val = (i%13)+1;
            deck->cards[i].face = to_string(deck->cards[i].val);
        }
    }
```

(Screenshot of added logic in *iniDeck)

Creating_Player_Structure

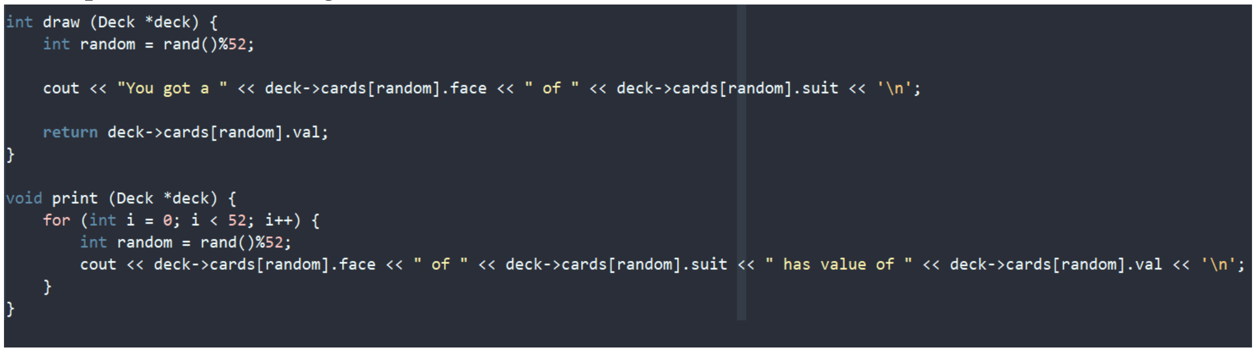
Having structures in this version of the project is the most beneficial when it came to being able to track player and dealer related things. Initially, I had four variables for the structure and it would expand later. I was able to track the total for the hands, wins, losses, and have a reference string for the player and the dealer. I could expand this in project 2 by having player's type a personal name and maybe even a betting system or multiple players (piloted by AI).



```cpp
struct Player {
    int hand = 0;
    int wins = 0;
    int loss = 0;
    string ref;
};
```

(Player.h file for Creating_Player_Structure)

In Creating_Player_Structure v2 the only thing that got added was a variable for any draws/pushes and creating a more refined draw function.

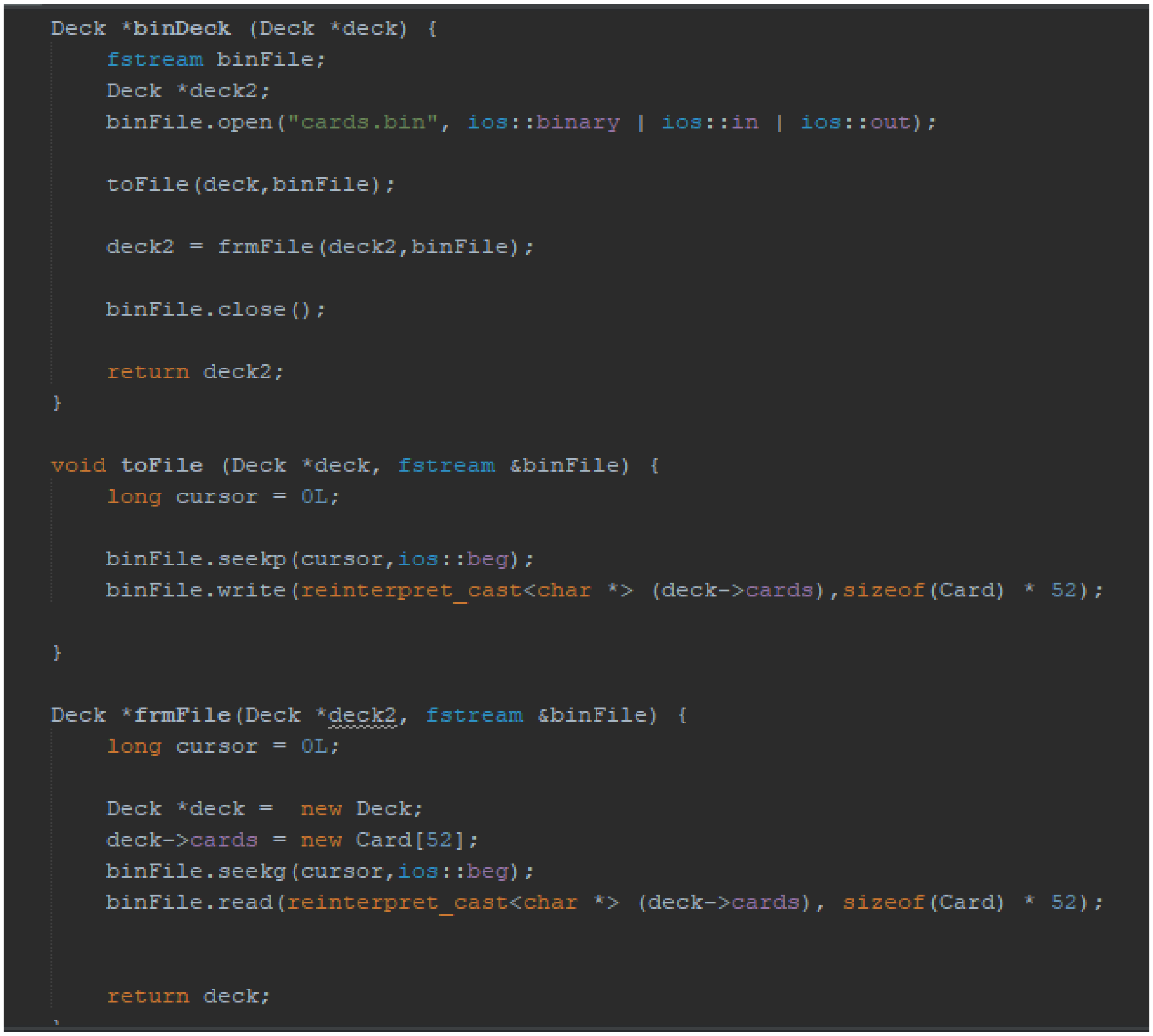


```cpp
int draw (Deck *deck) {
    int random = rand()%52;

    cout << "You got a " << deck->cards[random].face << " of " << deck->cards[random].suit << '\n';

    return deck->cards[random].val;
}

void print (Deck *deck) {
    for (int i = 0; i < 52; i++) {
        int random = rand()%52;
        cout << deck->cards[random].face << " of " << deck->cards[random].suit << " has value of " << deck->cards[random].val << '\n';
    }
}
```

(Two functions that were added and experimented with in Creating_Player_Structure_v2)

Binary_File_Deck_V1

This was probably the most frustrating part of developing this game. Messing with binary files is incredibly exhausting and personally, I dislike this the most. It multiple days to get it to work because of the structure is a pointer and not being sure what was happening. I got super frustrated multiple times because when I made this all I tried to do is I tried to write all the deck/card data into another deck pointer. I had the function write to a file then another one would read from the file and write it into another deck pointer. This was very frustrating to figure out.

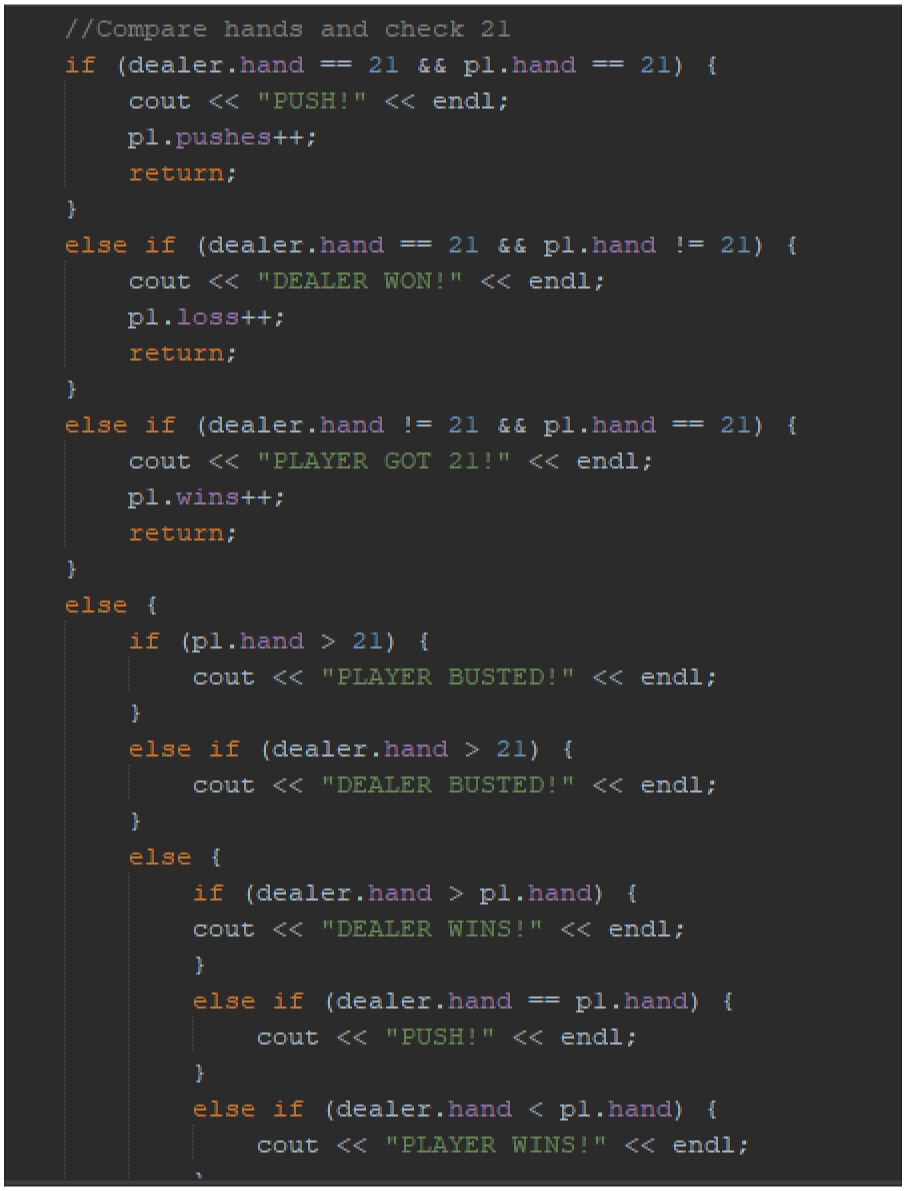


```
Deck *binDeck (Deck *deck) {
    fstream binFile;
    Deck *deck2;
    binFile.open("cards.bin", ios::binary | ios::in | ios::out);

    toFile(deck,binFile);

    deck2 = frmFile(deck2,binFile);

    binFile.close();

    return deck2;
}

void toFile (Deck *deck, fstream &binFile) {
    long cursor = 0L;

    binFile.seekp(cursor,ios::beg);
    binFile.write(reinterpret_cast<char *> (deck->cards),sizeof(Card) * 52);

}

Deck *frmFile(Deck *deck2, fstream &binFile) {
    long cursor = 0L;

    Deck *deck =  new Deck;
    deck->cards = new Card[52];
    binFile.seekg(cursor,ios::beg);
    binFile.read(reinterpret_cast<char *> (deck->cards), sizeof(Card) * 52);


    return deck;
```

(All of the binary functions)

Blackjack_Game_V1

I created the entire game in one version but it’s super sloppy and is really messy. Everything was in one game function and this version didn’t have a menu and didn’t have any binary file inclusion or enumeration.

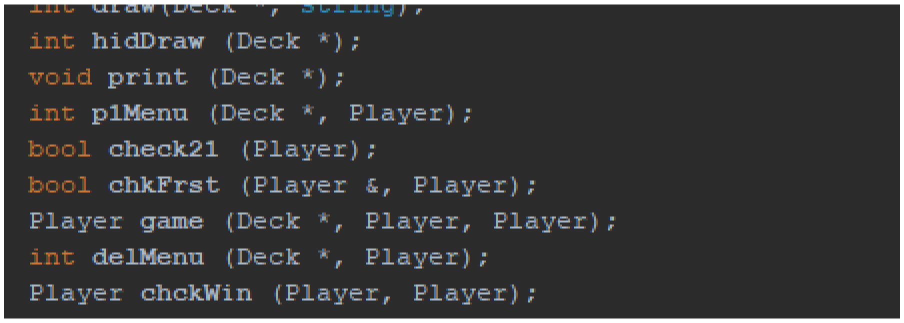


```
//Compare hands and check 21
if (dealer.hand == 21 && pl.hand == 21) {
    cout << "PUSH!" << endl;
    pl.pushes++;
    return;
}
else if (dealer.hand == 21 && pl.hand != 21) {
    cout << "DEALER WON!" << endl;
    pl.loss++;
    return;
}
else if (dealer.hand != 21 && pl.hand == 21) {
    cout << "PLAYER GOT 21!" << endl;
    pl.wins++;
    return;
}
else {
    if (pl.hand > 21) {
        cout << "PLAYER BUSTED!" << endl;
    }
    else if (dealer.hand > 21) {
        cout << "DEALER BUSTED!" << endl;
    }
    else {
        if (dealer.hand > pl.hand) {
        cout << "DEALER WINS!" << endl;
        }
        else if (dealer.hand == pl.hand) {
            cout << "PUSH!" << endl;
        }
        else if (dealer.hand < pl.hand) {
            cout << "PLAYER WINS!" << endl;
```

(Some of the messy and non-modularized code)

Blackjack_Game_V2

In this version I modularized all of the code in the game function. While there isn't much to write about this was a very important version of the game because it made everything much more readable, fixable, and functional. The game became a lot more clean and easier to debug. I also added a menu in this version and the ability to continuously replay games.

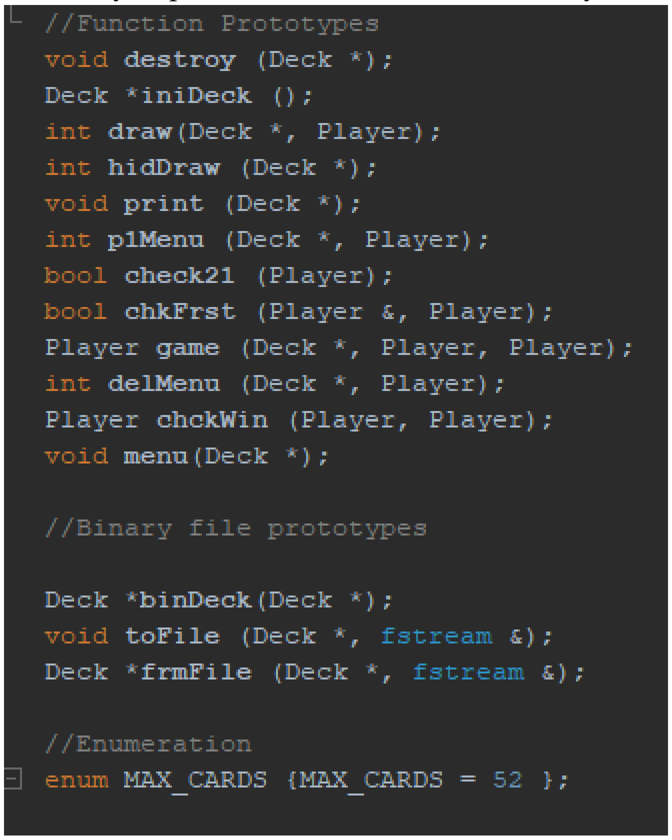


```
int hidDraw (Deck *);
void print (Deck *);
int p1Menu (Deck *, Player);
bool check21 (Player);
bool chkFrst (Player &, Player);
Player game (Deck *, Player, Player);
int delMenu (Deck *, Player);
Player chckWin (Player, Player);
```

(All of the new function prototypes that were added)

Blackjack_Game_V3

For V3 I basically just added what wasn't in V2 yet so, included the binary file, added logic for the Ace card so the game will automatically choose whether the Ace card is a '1' or an '11' based on which card the player needs. The binary files weren't added until last second to make sure nothing from the binary files would mess it up. I added a small enumeration variable to get that. I added a c-string variable pointer to get that as well. I also expanded the menu to include the ability to print all the cards from the binary file deck.



```
//Function Prototypes
void destroy (Deck *);
Deck *iniDeck ();
int draw(Deck *, Player);
int hidDraw (Deck *);
void print (Deck *);
int p1Menu (Deck *, Player);
bool check21 (Player);
bool chkFrst (Player &, Player);
Player game (Deck *, Player, Player);
int delMenu (Deck *, Player);
Player chckWin (Player, Player);
void menu(Deck *);

//Binary file prototypes

Deck *binDeck(Deck *);
void toFile (Deck *, fstream &);
Deck *frmFile (Deck *, fstream &);

//Enumeration
enum MAX_CARDS {MAX_CARDS = 52 };
```

(List of all the functions and enumeration)

Psuedo Code

```
Initilize deck and deck2
Read and write from binary file for deck2

Display welcome message and send deck 2

If menu equals 2
    Display the rules

If menu equals 3
    Print all the cards from deck 2

If menu equals 4
    Exit program

If menu equals 1

    Begin the game
    Draw 2 cards for player
    Draw 2 cards for Dealer
    Reveal first card for dealer

    If either people have 21
        Announce winner
        end game
    else if player wants to hit
        draw another card until done
    else
        dealer draws until hand is 17 or more

    compare each other's hand
        if player won
            print winner message, add 1 to total wins
        else if player lost
            print loser message, add 1 total losses
        else if push
            print push message, add 1 to draw


    ask player if they wish to play again and loop beginning with menu
```

FlowChart

Please check file “flowchartproject1.drawio.pdf” in the BlackjackProject1.zip to find the flowchart.

Check-off Sheet

Please check file “checkoffsheet.pdf” to find the Check-off sheet.

Github Link

Code

```
/*
 * File:   main.cpp
 * Author: Christian Fuentes
 * Created on November 3, 2021. 2:22 PM
 * Purpose:  Game with binary
 * Modularize and optimize the game
 * (Really terrible version 1 in my opinion....)
 * Create logic for Ace card.
 * Added menu
 */

//System Libraries
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <fstream>

using namespace std;

//User Libraries
#include "Deck.h"
#include "Player.h"
//Global Constants - No Global Variables

//Function Prototypes
void destroy (Deck *);
Deck *iniDeck ();
int draw(Deck *, Player);
int hidDraw (Deck *);
```

```
void print (Deck *);
int p1Menu (Deck *, Player);
bool check21 (Player);
bool chkFrst (Player &, Player);
Player game (Deck *, Player, Player);
int delMenu (Deck *, Player);
Player chckWin (Player, Player);
void menu(Deck *);

//Binary file prototypes

Deck *binDeck(Deck *);
void toFile (Deck *, fstream &);
Deck *frmFile (Deck *, fstream &);

//Enumeration
enum MAX_CARDS {MAX_CARDS = 52 };

//Execution Begins Here
int main(int argc, char** argv) {
   //Set random seed
   srand (static_cast<unsigned int> (time(NULL)));
   //Declare Variable Data Types and Constants
   Deck *deck = iniDeck();
   Deck *deck2 = binDeck(deck);
   Player p1,
          dealer;
   char again = 'y';
   //Initialize Variables


   //Process or map Inputs to Outputs
   menu(deck2);
   do {
      p1 = game (deck, p1, dealer);
      cout << "Would you like to play again? Enter y or Y.\n";
      cin >> again;
   }while (again == 'y' || again == 'Y');


   //Display Outputs
   destroy (deck);
   delete deck2;
   //Exit stage right!
```

```
    return 0;
}

void destroy (Deck *deck) {
    delete []deck->cards;
    delete deck;
}

Deck *iniDeck () {
    Deck *deck = new Deck;


    for (int i = 0; i < MAX_CARDS; i++) {
        switch (i%4) {
            case 0: deck->cards[i].suit = "Hearts"; break;
            case 1: deck->cards[i].suit = "Diamonds"; break;
            case 2: deck->cards[i].suit = "Spades"; break;
            case 3: deck->cards[i].suit = "Clubs"; break;
        }
        if ((i%13)+1 > 10) {
            switch ((i%13)+1) {
                case 11: deck->cards[i].face = "Jack"; break;
                case 12: deck->cards[i].face = "Queen"; break;
                case 13: deck->cards[i].face = "King"; break;
            }
            deck->cards[i].val = 10;
        }
        else if ((i%13)+1 == 1) {
            deck->cards[i].face  = "Ace";
            deck->cards[i].val = 1;
        }
        else {
            deck->cards[i].val = (i%13)+1;
            deck->cards[i].face = to_string(deck->cards[i].val);
        }
    }
    return deck;
}

int draw (Deck *deck, Player player) {
    int random = rand()%52;

    cout << player.ref << " got a " << deck->cards[random].face << " of " <<
deck->cards[random].suit << '\n'; //Open draw for player and dealer
```

```
    if (deck->cards[random].val == 1 && player.hand <= 10) {                                   //Ace logic
        return 11;
    }



    return deck->cards[random].val;
}

int hidDraw (Deck *deck) {                                                          //Hidden draw for
dealer
    int random = rand()%52;

    if (deck->cards[random].val == 1) {
        return 11;
    }

    return deck->cards[random].val;
}

void print (Deck *deck) {                                                           //Print for the deck
    for (int i = 0; i < 52; i++) {
        cout << deck->cards[i].face << " of " << deck->cards[i].suit << " has value of " <<
deck->cards[i].val << '\n';
    }
}


Player game (Deck *deck, Player p1, Player dealer) {
    //Initialize player and dealer
    p1.hand = 0;
    p1.ref = "You";

    dealer.hand = 0;
    dealer.ref = "Dealer";
    //Initialize player draw
    p1.hand += draw(deck,p1);
    p1.hand += draw(deck,p1);

    cout << "Your hand is " << p1.hand << '\n';

    //Initialize dealer hand
    dealer.hand += draw(deck,dealer);
    dealer.hand += hidDraw(deck);                    //Doesn't print out this card

    //Check if either player got ace from first turn.
    if (chkFrst(p1,dealer) == true) {
        return p1;
    }

    //p1menu will ask player if stand or hit.
    p1.hand = p1Menu(deck, p1);

    cout << "\nDealer hand is " << dealer.hand << '\n' << '\n';
    dealer.hand = delMenu(deck, dealer);

    cout << "\nDealer hand is " << dealer.hand << '\n';

    //Compare hands and check 21
    p1 = chckWin(p1,dealer);

    return p1;
}


//Using this to check if anyone got 21 within the first cards that were drawn
//If anyone did the function will return true and in the game function
//The game function will return and exit the game announcing the winner
bool chkFrst (Player &p1, Player dealer) {
    //Check 21 for the player.
    if (check21(p1) == true) {
        p1.frstTrn = true;
    }
    //Check 21 for dealer
    if (check21(dealer) == true) {
        dealer.frstTrn = true;
    }
    //Fully check all possibilities for player or dealer getting 21
    if (dealer.frstTrn == true && p1.frstTrn == true) {
        cout << "PUSH!" << endl;
        p1.pushes++;
        return true;
    }
    else if (dealer.frstTrn == true && p1.frstTrn == false) {
        cout << "DEALER GOT 21!" << endl;
        p1.loss++;
        return true;
    }
    else if (dealer.frstTrn == false && p1.frstTrn == true) {
        cout << "PLAYER GOT 21!" << endl;
        p1.wins++;
        return true;
    }
    return false;
}


bool check21 (Player player) {
    if (player.hand == 21) {
        return true;
    }
    return false;
}


Player chckWin (Player p1, Player dealer) {
    if (dealer.hand == 21 && p1.hand == 21) {          //dealer compare for 21 and player 21
        cout << "PUSH!" << endl;
        p1.pushes++;
        return p1;
    }
    else if (dealer.hand == 21 && p1.hand != 21) {     //dealer check for 21
        cout << "DEALER GOT 21!" << endl;
        p1.loss++;
        return p1;
    }
    else if (dealer.hand != 21 && p1.hand == 21) {     //player check  for 21
        cout << "PLAYER GOT 21!" << endl;
        p1.wins++;
        return p1;
    }
    else {
        if (p1.hand > 21) {                            //check for player bust
            cout << "PLAYER BUSTED!" << endl;
            p1.loss++;
            return p1;
        }
        else if (dealer.hand > 21) {                   //check for dealer bust
            cout << "DEALER BUSTED!" << endl;
            p1.wins++;
            return p1;
        }
        else {
            if (dealer.hand > p1.hand) {              //check if dealer greater
                cout << "DEALER WINS!" << endl;
                p1.loss++;
                return p1;
            }
            else if (dealer.hand == p1.hand) {        //check push
                cout << "PUSH!" << endl;
                p1.pushes++;
                return p1;
            }
            else if (dealer.hand < p1.hand) {         //check player greater
                cout << "PLAYER WINS!" << endl;
                p1.wins++;
                return p1;
            }
        }
    }
    return p1;                                        //return to keep track of player.
}


int p1Menu (Deck *deck, Player p1) {
    char choice;
    do {
        cout << "Press 1 to hit." << endl;
        cout << "Press 2 to stand." << endl;
        cin>>choice;
        if (choice == '1' && p1.hand < 21) {
            p1.hand += draw(deck,p1);
        }
        cout << "Your hand is " << p1.hand << endl;
    }while (choice == '1' && p1.hand < 21);
    return p1.hand;
}

int delMenu(Deck *deck, Player dealer) {
    while (dealer.hand < 17) {
        dealer.hand += draw(deck,dealer);
        cout << "Dealer hand is now " << dealer.hand << '\n';
    }
    return dealer.hand;
```

```
}

void menu (Deck *deck2) {
    char *choice = new char [32];                                  //random size, doesn't matter
will be deallocated later :)
    do {
        cout << "Welcome to Christian's Blackjack!" << '\n';
        cout << "Press 1 to play Blackjack.\n"
             << "Press 2 to see how to play Blackjack.\n"
             << "Press 3 to print binary file version of deck.\nPress 4 to exit.\n";
        cin>>choice;
        if (*choice == '2') {
            cout << "The goal of blackjack is to get a hand equal to 21." <<
                        "Each card is worth its numerical value, face cards " <<
                        "are worth 10, and ace is worth either 1 or 11.\n" <<
                        "Both the player and the dealer start with 2 cards " <<
                        "and the dealer flips " <<
                        "only 1 card up to show to the player. The player must " <<
                        "then decide if they want to hit (draw)\nor stand (not " <<
                        "draw).Press 1 to hit and 2 to stand.\n";
        }
        else if (*choice == '3') {
            print(deck2);
        }
        else if (*choice == '4') {
            delete []choice;
            exit(0);
        }
        else if (*choice > '5' || *choice < '1') {
            cout << "Please enter  valid number\n";
        }
    }while (*choice != '1');
    delete []choice;
}

Deck *binDeck (Deck *deck) {
    fstream binFile;
    Deck *deck2;
    binFile.open("card.bin", ios::binary | ios::in | ios::out);

    toFile(deck,binFile);

    deck2 = frmFile(deck2,binFile);
    binFile.close();

    return deck2;
}

void toFile (Deck *deck, fstream &binFile) {
    long cursor = 0L;

    binFile.seekp(cursor,ios::beg);
    binFile.write(reinterpret_cast<char *> (deck->cards),sizeof(Card) * 52);

}

Deck *frmFile(Deck *deck2, fstream &binFile) {
    long cursor = 0L;

    Deck *deck =  new Deck;
    binFile.seekg(cursor,ios::beg);
    binFile.read(reinterpret_cast<char *> (deck->cards), sizeof(Card) * 52);


    return deck;
}
```