Project 2

<Blackjack>

CIS-5-43358

Name: Christian Fuentes

Date: June 6, 2021

Introduction

Title: Blackjack Game

Blackjack is a fairly simple game, the goal is to get as close to the number 21 as possible if someone gets 21 it is called a "Blackjack." Ace is worth either 1 or 11, the faces are worth 10 each, and all the other cards are worth their numeric value. The game first starts off with both the player and the dealer drawing 2 cards but the dealer reveals one of his cards. The player can either hit (draw) or stand (keep their hand) until their hand is 21 or they bust (go over 21). The dealer will reveal his 2nd card and if it's less than 17 he will draw until it's over 17 or he busts. If neither player busts then the winner is whoever is closest to 21, if the dealer and the player have an equal hand then it's a push (draw).

Summary

Project size: 494 Lines (final version)

The concepts in this project are covered from everything we learned in CSC-5. I did introduce a betting system in this version compared to project 1. This one does include functions, vectors, arrays, searching, and sorting. I did not include splitting and double-down in my program of Blackjack nor multiple players, only dealer, and 1 player.

I worked on this over the span of a week but I worked about 3-8 hours every day on the code and or ideas. Creating this project was very interesting with functions because compared to last time I feel like I had to really try and predict where everything needed to go and everything that needed to be done before I even got to program the game.

Development

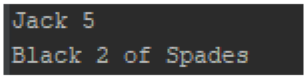
(Not done in chronological order as I made certain side programs when I needed to test things/adjustments)

Constructing_Deck_Array_V1

The first thing I worked on was making a deck array. This is a great example of my lack of knowledge on array's heading in. This program didn't take advantage of the fact that I was working with arrays and instead, I made an array that contained strings of characters used for cards ex: "Ace of Spades" was just a string of "Ace" and "Spades" with "of" being outputted but not in the array. This was a terrible use of array's which wouldn't be fixed until later.

Constructing_Deck_Array_V2

I began to work on my introduction to vectors and playing around with them. I had not fixed my deck array as I thought it was good until the later development. I made a vector array that was similar to my deck array but instead just used vectors. I used this program to get more comfortable with vectors.

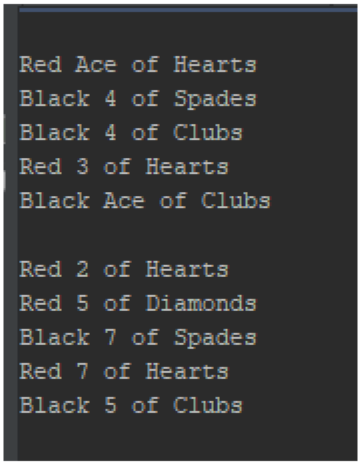


```
Jack 5
Black 2 of Spades
```

(example output of Constructing_Deck_Array_V2)

Constucting_Deck_Array_V3

In this one, I focused on making a shuffle and printing deck function. I used 3 arrays within a function to create a deck array. I made 3 string arrays consisting of the color, suit, and face of the card and added all those strings together into one other string array. This was a terrible approach to making a deck but it also helped me learn how to better work with functions.



```
Red Ace of Hearts
Black 4 of Spades
Black 4 of Clubs
Red 3 of Hearts
Black Ace of Clubs

Red 2 of Hearts
Red 5 of Diamonds
Black 7 of Spades
Red 7 of Hearts
Black 5 of Clubs
```

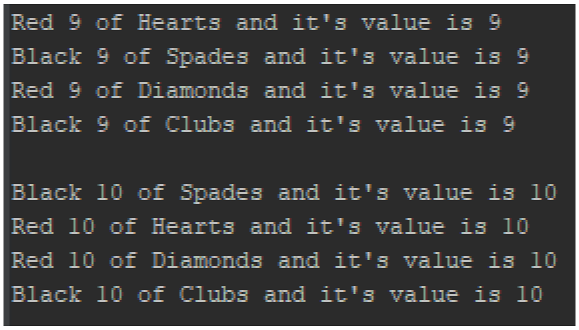
(Example output of the deck being shuffled)

Constructing_Deck_Array_V4

In this version, I re-created the deck array and used it for vectors with the same approach. It was very terrible to make 3 vectors and make it into 1 big vector containing the deck.
(File got erased, so I can’t post screenshots)

Constructing_Deck_Array_V5_Searching_Sorting

I created a bubble and selection sort algorithm to see if I could implement it using my deck. I then used the same algorithm but created more functions to test if the same code could work using vectors and it did. I also did a little test to make sure that the value’s matched with the card face string name. I also put in a linear search but it got deleted.

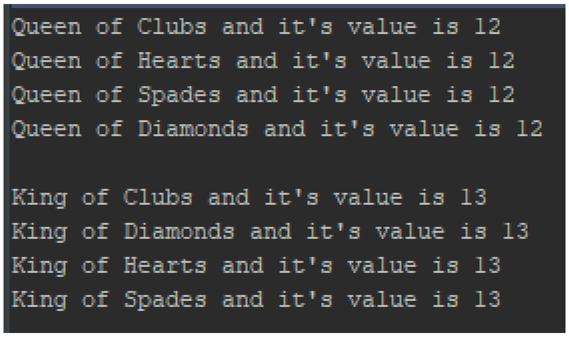


```
Red 9 of Hearts and it's value is 9
Black 9 of Spades and it's value is 9
Red 9 of Diamonds and it's value is 9
Black 9 of Clubs and it's value is 9

Black 10 of Spades and it's value is 10
Red 10 of Hearts and it's value is 10
Red 10 of Diamonds and it's value is 10
Black 10 of Clubs and it's value is 10
```

(Deck sorted and shuffle)

Constructing_Deck_Array_V6_Fixing_Vectors

Now, this is the most pivotal part of the project where I learned how to correctly use arrays and vectors. I took out the color string array and vector and created a vector that used both the face and suit and generated them. I created another vector and array that was parallel with the string version of themselves. Now I could access each vector/array and not have to worry about how to make the string match the integer version of themselves. It doesn't seem that complicated but it took me a while to figure this out!

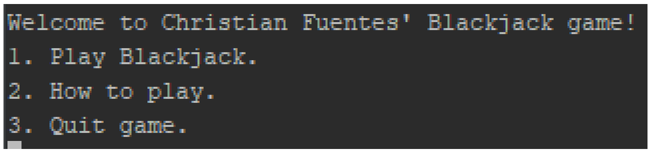


```
Queen of Clubs and it's value is 12
Queen of Hearts and it's value is 12
Queen of Spades and it's value is 12
Queen of Diamonds and it's value is 12

King of Clubs and it's value is 13
King of Diamonds and it's value is 13
King of Hearts and it's value is 13
King of Spades and it's value is 13
```

(Sorted vector deck and its numerical value inside the integer version of the vector.)

Menu

The same menu I used in project 1 and re-used in project 2. I just made it a function here to see if it could function all the same. I added 1 more feature to the menu in the final game but we'll get to that later.

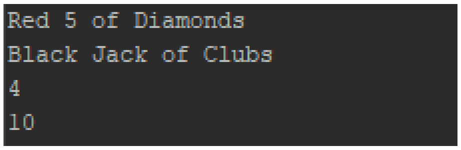


```
Welcome to Christian Fuentes' Blackjack game!
1. Play Blackjack.
2. How to play.
3. Quit game.
```

(What the menu looks like)

Player_Dealer_Hand_Functions_V1

I created this early in development as this didn't come before Constructing_Deck_Array_5 but it was just to see how I would re-create the functioning of player and dealer hand functions that I created from project 1. I also wanted to see if I wanted to make the hand functions an int or a void.



```
Red 5 of Diamonds
Black Jack of Clubs
4
10
```

(As you can see it isn't matching the numeric value with the string of text. I would fix this in Constructing_Deck_Array_V6_Fixing_Vectors)

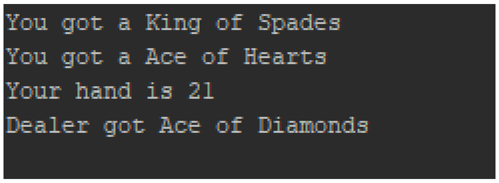
Blackjack_Game_V5
(Versions 1-4 can be found in Project 1, I just continued the naming from project 1. This is the first version for project 2).

I basically just moved all the functions from the previous programs into here (not vectors yet). I moved in the getCard functions, shuffle, prntDeck, and menu function. Overall this was just me moving everything I had discovered so far (This was before Constructing_Deck_Array_V6_Fixing_Vectors).

```
2
```

(All it did was print player hand function)

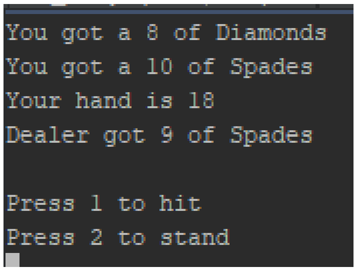
Blackjack_Game_V6
Here I added vectors and this was after Constructing_Deck_Array_V6_Fixing_Vectors. I worked out all the logic for the actual game function and added more functions to check for blackjack but I didn't add the wins or loss based on who had a higher hand and who busted.



```
You got a King of Spades
You got a Ace of Hearts
Your hand is 21
Dealer got Ace of Diamonds
```

(Example output)

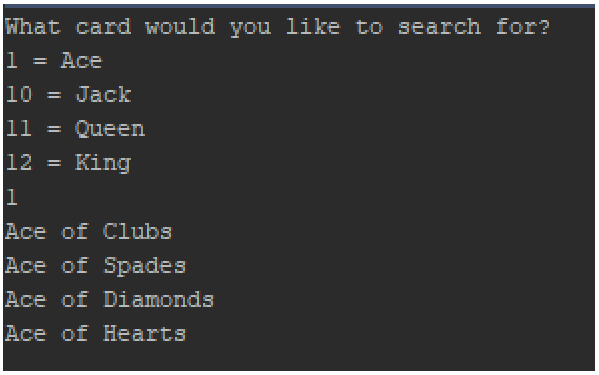
Blackjack_Game_V7
I added winning and losing based on who was closer to 21. I added to see if anyone busted and the feature of standing or hitting. The game also didn't loop yet but version 8 will be the final one and this was a smaller update.



```
You got a 8 of Diamonds
You got a 10 of Spades
Your hand is 18
Dealer got 9 of Spades

Press 1 to hit
Press 2 to stand
```

(New example output)

Blackjack_Game_V8
I added the ability to do a linear search for a certain card in the deck and print all the cards of that face value. I added the menu to work in the game and added the option to do the linear search in the menu. I added the ability to loop the game. The ability to bet was also introduced here along with file i/o of wins and win rate. Here I also made sure the game sorts the decks at the end of a game and shuffles again when it starts. I also tried to organize the code a little more by array functions vs vector functions vs game functions.



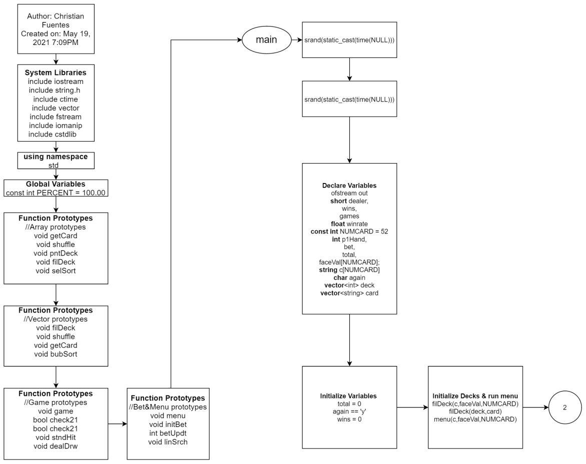
```
What card would you like to search for?
1 = Ace
10 = Jack
11 = Queen
12 = King
1
Ace of Clubs
Ace of Spades
Ace of Diamonds
Ace of Hearts
```

(Example of searching for a card)

Pseudocode + Flowchart

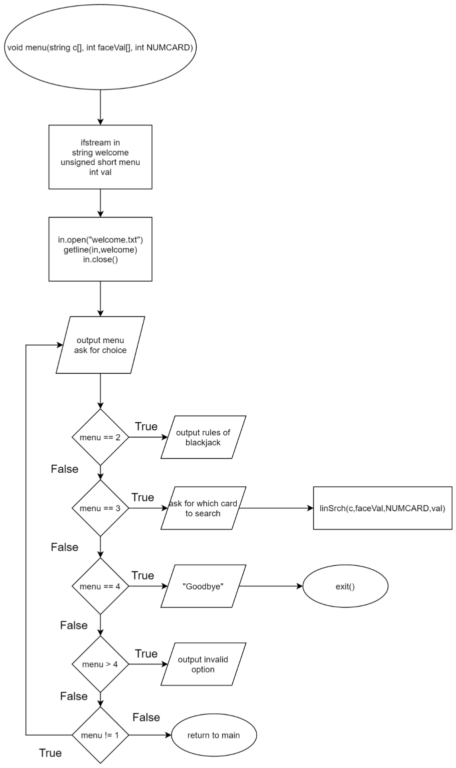
The full flowchart will be in the Project 2 folder and will be in its complete form. This will be pseudocode complimented by a flowchart that copies the actual code in logical order. The flowchart on Github and the zip file will be the completed flowchart for each function and their section (main, game, array, vector, menu, and betting). It will also **NOT** include some flowcharts (as they might be seen as less important) such as the linear search for certain cards in the deck. Please refer to [GitHub](github.com/UtotingPuwet) (github.com/UtotingPuwet) or project2.zip to see full flowcharts.

```
Include all necessary system libraries
Use namespace std
Initialize global variable percent = 100
Initialize all function prototypes

Main ()
        srand(static_cast(time(NULL)))
        Declare all variables
        Initialize variables

        Initialize decks with
        filDeck functions
        Go/run the menu
```



Menu options consist of
Play the game,
Learn the rules,
Search deck for a specific card,
Or quit program
We will verify all user input throughout
This section of the menu
When the choice doesn't equal 2-4 we will begin
The game

We will initialize the total amount of money
User is willing to spend
And then ask for their bet which
Can't be over $50,000

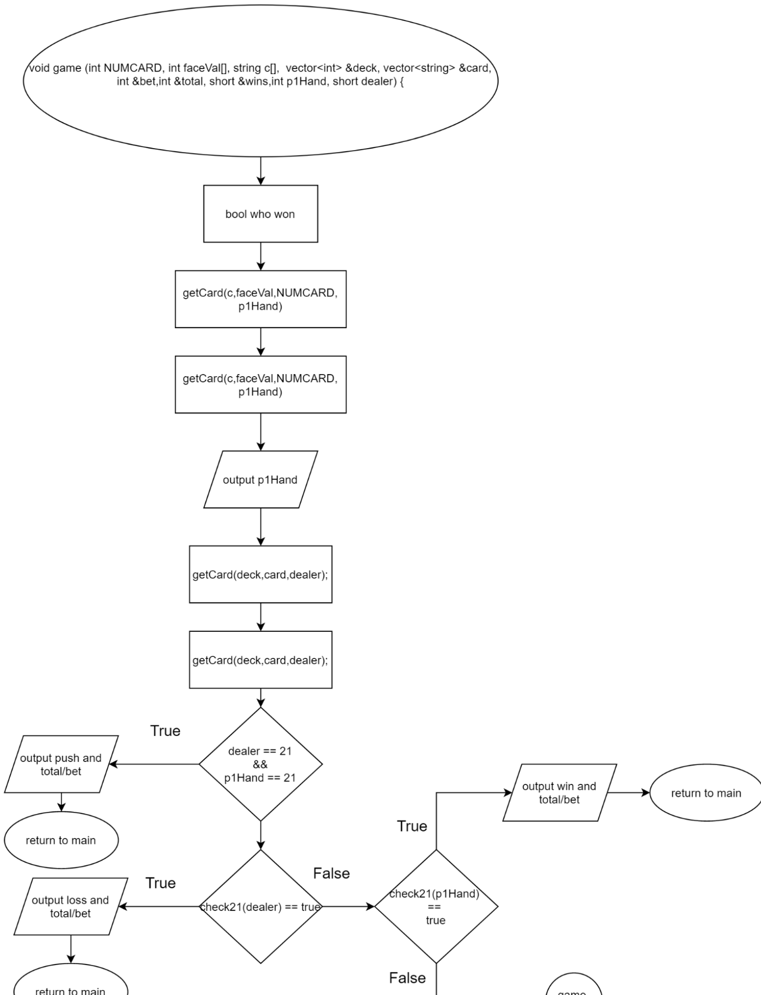


Begin game function

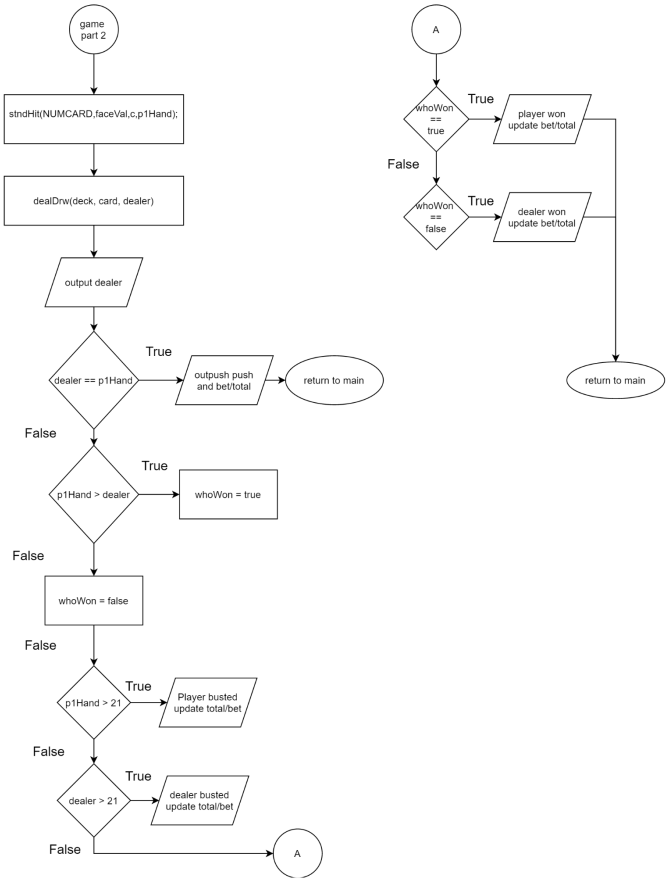
Initialize bool whoWon
Grab 2 cards for the player
Output player's hand

Get a card for the dealer
Print only the first card

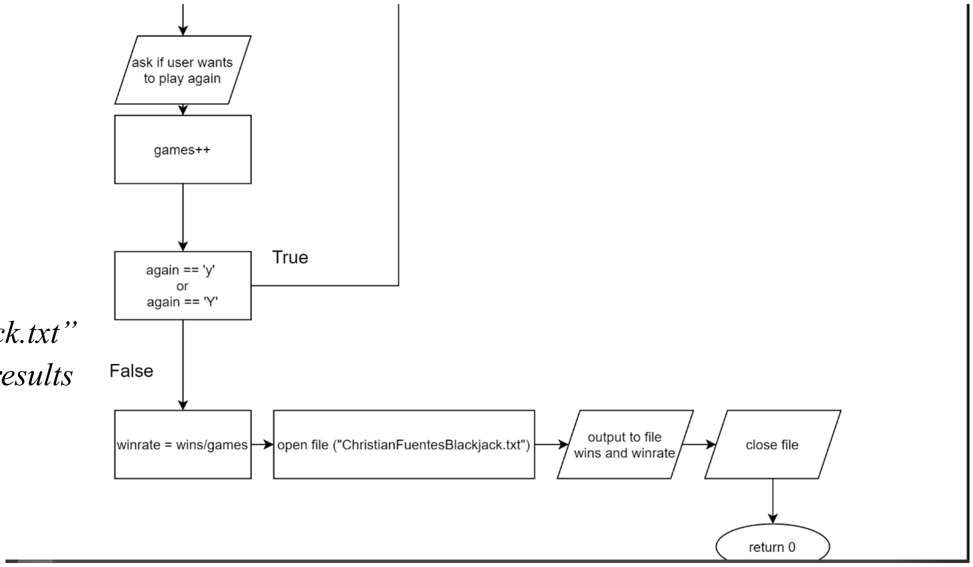
Run stndHit function
If player wants to hit
Which is the "1" choice
Player draws a card
Once player is over 21 or
Chooses "2" exit stndHit


void game (int NUMCARD, int faceVal[], string c[], vector<int> &deck, vector<string> &card, int &bet,int &total, short &wins,int p1Hand, short dealer) {
bool who won
getCard(c,faceVal,NUMCARD, p1Hand)
getCard(c,faceVal,NUMCARD, p1Hand)
output p1Hand
getCard(deck,card,dealer);
getCard(deck,card,dealer);
True
dealer == 21
&&
p1Hand == 21
output push and total/bet
return to main
True
output loss and total/bet
check21(dealer) == true
return to main
False
check21(p1Hand)
==
true
True
output win and total/bet
return to main
False
game

```
Draw cards for the dealer
        Until the dealer is
        Over 17 or busts.

Run check21 for both player
        And the dealer
        If either of them are
        21 then update bet
        And return to main

If either are over 21
        Output who busted
        And update bet
        Accordingly

If neither are over 21
        Check who is
        Closest to 21
Output the winner
        Update bet
```



```
Go back to main

Add 1 to games to track
        Number of games played

Ask if user wants to play again
        If yes then loop
If not then open file
        "ChrsitianFuentesBlackjack.txt"
        Output winrate and game results
        There
```


ask if user wants to play again
games++
again == 'y' or again == 'Y'
True
False
winrate = wins/games
open file ("ChristianFuentesBlackjack.txt")
output to file wins and winrate
close file
return 0

Other important functions

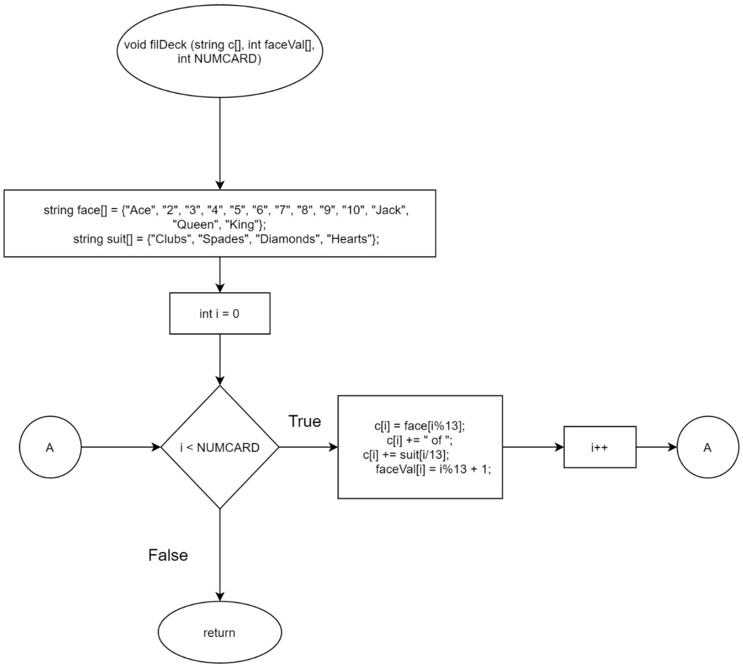
filDeck function

Initialize string array face and suit

Fill another string array called c
By face%13 to get all faces and
All suit by 13 to get all suits

Fill int array faceVal but i%13+1 as
It will be a parallel array to
String array c

Vector is the same


void filDeck (string c[], int faceVal[], int NUMCARD)
string face[] = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King"};
string suit[] = {"Clubs", "Spades", "Diamonds", "Hearts"};
int i = 0
A
i < NUMCARD
True
c[i] = face[i%13];
c[i] += " of ";
c[i] += suit[i/13];
faceVal[i] = i%13 + 1;
i++
A
False
return

getCard function

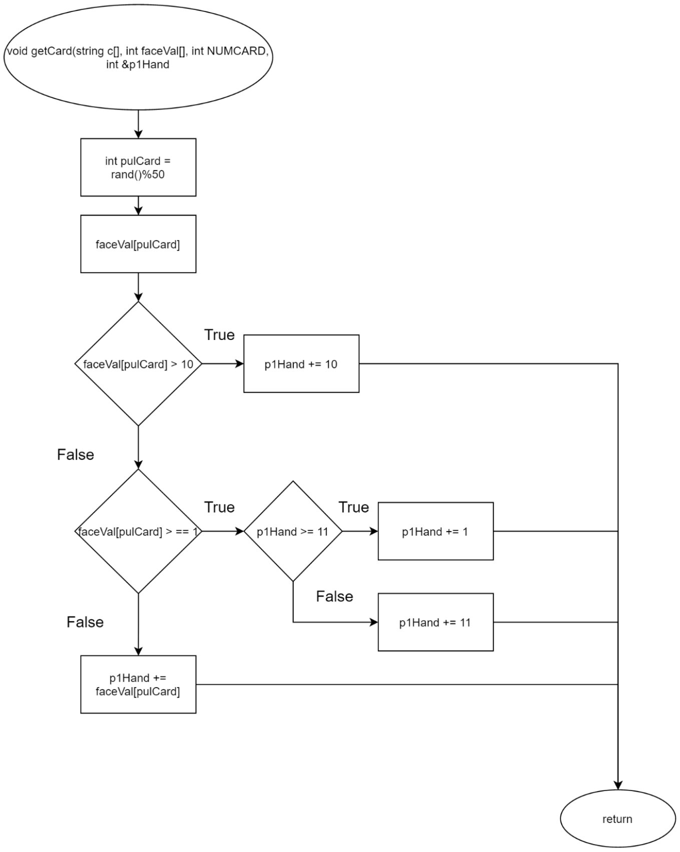
Get a random number and
Go to that index of faceVal

If faceVal > 10 add 10 to player's hand

If faceVal == 1 add either
11 if player hand is <11
1 if player hand is >11

Output player's card

This is the same as the dealer's function



Code

```
/*
   Author: Christian Fuentes
   Date:   May 31 2021, 5:55 PM
   Purpose:Re-make Game through functions and arrays and vectors.
   Version:4
 *
 * Adding a betting system
 */

//System Libraries
#include <iostream>    //Input/Output Library
#include <string.h>    //String Library
#include <ctime>       //Time Library
#include <vector>      //Vector Library
#include <fstream>     //File input/output
#include <iomanip>     //Manipulate Library
#include <cstdlib>     //
using namespace std;   //Library Name-space

//User Libraries

//Global/Universal Constants -- No Global Variables
//Science, Math, Conversions, Higher Dimensioned constants only
const float PERCENT = 100.00;
//Function Prototypes
//---------------------------------------------------------------------------------------------------------------
----------------------------
//All below will be vector function prototypes
//---------------------------------------------------------------------------------------------------------------
----------------------------

void filDeck(vector<int> &, vector<string> &);          //fill the vector deck
void getCard(vector<int> &, vector<string> &, short &); //draw cards from the vector deck. Will
be used for dealer
void shuffle(vector<int> &, vector<string> &);          //shuffle the vector deck
void bubSort(vector<int> &, vector<string> &);          //sort the vector with bubble sort algorithm
//---------------------------------------------------------------------------------------------------------------
----------------------------
```

```
//All below will be array function prototypes
//------------------------------------------------------------------------------------------------------------
----------------------------
void getCard(string [], int [], int, int&);            //will be used to draw cards for player
void shuffle(string [], int [], int);                  //shuffle the array deck
void pntDeck(string [], int [], int);                  //print the array deck
void filDeck(string [], int [], int);                  //fill the array deck
void selSort(string [], int [], int);                  //sort the array with select sort algorithm
//------------------------------------------------------------------------------------------------------------
----------------------------
//All below will be game function  prototypes
//------------------------------------------------------------------------------------------------------------
----------------------------

void game (int, int[], string [], vector<int> &, vector<string> &, int &, int &,short &,int=0,
short=0);//play blackjack
bool check21 (short);                                  //check 21 for dealer
bool check21 (int);                                    //check 21 for player
void stndHit (int, int [], string c[], int &);         //player stand or hit or double down
void dealDrw(vector<int> &, vector<string> &, short &); //dealer auto draw if under 17

//------------------------------------------------------------------------------------------------------------
----------------------------
//All below will be menu/betting function  prototypes
//------------------------------------------------------------------------------------------------------------
----------------------------

void menu(string [], int [], int);                     //just a welcome message :]
void initBet(int &, int &);                            //initialize the bet money
int betUpdt (int, int &, int);                         //update bet depend in number. 0 = lose, 1 = win, 2 =
push, 3 = blackjack
void linSrch (string [], int [], int, int);            //search for certain cards
//Execution Begins Here
int main(int argc, char** argv) {
   //Set the Random number seed
   srand(static_cast<unsigned int>(time(NULL)));

   //Declare variables
   ofstream out;
   short dealer,                                       //variable for dealer hand
```

```
        wins,                          //track wins
        games;                          //track games
   float winrate;                        //calculate winrate by wins/games
   const int NUMCARD = 52;                        //size of a standard playing deck (this is  used to
initialize array size)
   int p1Hand,                           //player 1 hand
      bet,                            //how much player 1 is betting
      total,                          //total that the player started with
      faceVal[NUMCARD];                        //integer value of each card
   string c[NUMCARD];                         //string array for each card
   char again;                          //variable to see if player wants to play again
   vector<int> deck;                         //vector for face value of each card in the deck
   vector<string> card;                       //vector for string for each card
   //Initialize variables
   total = 0;
   again == 'y';
   wins = 0;
   //Initialize decks
   filDeck(c,faceVal,NUMCARD);
   filDeck(deck,card);
   menu(c,faceVal,NUMCARD);
   do {

      shuffle(c,faceVal,NUMCARD);
      shuffle(deck,card);

      //Initialize bet/total
      initBet(bet,total);

      //Begin game
      game(NUMCARD,faceVal,c,deck,card,bet,total,wins);

      //Sort deck after game is finished
      selSort(c,faceVal,NUMCARD);
      bubSort(deck,card);
      cout << "Would you like to play again? Y for yes. N for no.\n";
      games++;
      cin>>again;
   } while (again == 'y' || again == 'Y');
```

```
    //Display your initial conditions as well as outputs.
    winrate = wins/games;
    out.open("ChristianFuentesBlackJack.txt",ios::out);
    out << "Your wins for this session are   : " << static_cast<int>(wins) << '\n';
    out << "Your winrate for this session is :"  << fixed << setprecision(2) <<
abs(winrate*PERCENT) <<"%\n";
    out.close();
    //Exit stage right
    return 0;
}

//------------------------------------------------------------------------------------------------------------
------------------------------
//functions below will belong to anything with vectors.
//------------------------------------------------------------------------------------------------------------
------------------------------

void filDeck(vector<int> &deck, vector<string> &card) {
    vector<string> face = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen",
"King"};
    vector<string> suit = {"Clubs", "Spades", "Diamonds", "Hearts"};

    for (int i = 0; i < 52; i++) {
        card.push_back(face[i%13] + " of " + suit[i/13]);
        deck.push_back(i%13+1);
    }
    return;
}

void getCard(vector<int> &deck, vector<string> &card, short &dealer) {
    int randVal = rand()%13 + 1;

    if (dealer == 0) {
        cout << "Dealer got " << card[randVal] << '\n';
    }
    if (deck[randVal] > 10) {
        dealer += 10;
    }
    else if (deck[randVal] == 1) {
        if (dealer >= 11) {
            dealer += 1;
        }
        else {
            dealer += 11;
        }
    }
    else {
        dealer += deck[randVal];
    }
    return;
}

void  shuffle (vector<int> &deck, vector<string> &card) {
    for (int shfl = 0; shfl < 7; shfl++) {
        for (int i = 0; i < 52; i++) {
            int randVal = rand()%52;
            string temp = card[i];
            card[i] = card[randVal];
            card[randVal] = temp;

            int itemp = deck[i];
            deck[i] = deck[randVal];
            deck[randVal] = itemp;
        }
    }
    return;
}

void bubSort(vector<int> &deck, vector<string> &card) {
    bool swap;
    do {

        swap = false;
        for (int i = 0; i < 52-1; i++) {
            if (deck[i] > deck[i+1]) {
                int temp = deck[i];
                deck[i] = deck[i+1];
                deck[i+1] = temp;
                swap = true;
                string tempS = card[i];
                card[i] = card[i+1];
                card[i+1] = tempS;
            }
        }
    }while(swap);
    return;
}

//------------------------------------------------------------------------------------------------------------------------
-------------------------------
//functions below will belong to anything with arrays.
//------------------------------------------------------------------------------------------------------------------------
-------------------------------

void getCard(string c[], int faceVal[], int NUMCARD, int &p1Hand) {

    //declare variables
    int pulCard = rand()%52+1;
    //initialize and output
    faceVal[pulCard];

    //checking if player gets jack,king,queen
    if (faceVal[pulCard] > 10) {
        p1Hand+= 10;
    }
    //checking if player gets an ace and to see if hand is more than or equal to 11
    else if (faceVal[pulCard] == 1){
        if (p1Hand >= 11) {
            p1Hand += 1;
        }
        else p1Hand += 11;
    }
    else {
        p1Hand+= faceVal[pulCard];
    }

    cout << "You got a " << c[pulCard] << '\n';
    return;
```

```
}


void filDeck (string c[], int faceVal[], int NUMCARD) {
    string face[] = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King"};
    string suit[] = {"Clubs", "Spades", "Diamonds", "Hearts"};
    for (int i = 0; i < NUMCARD; i++) {
            c[i] = face[i%13];
            c[i] += " of ";
            c[i] += suit[i/13];

            faceVal[i] = i%13 + 1;
    }
    return;
}

void pntDeck (string c[], int faceVal[], int NUMCARD) {
    int count = 0;
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 5; j++) {
            cout << c[count] << "and it's value is " << faceVal[count] << endl;
            count++;
        }
        cout << endl;
    }
   return;
}

void shuffle (string c[], int faceVal[], int NUMCARD) {
    for (int shfl = 0; shfl < 7; shfl++) {
        for (int i = 0; i < NUMCARD; i++) {
            int randVal = rand()%NUMCARD;
            string temp = c[i];
            c[i] = c[randVal];
            c[randVal] = temp;

            int itemp  = faceVal[i];
            faceVal[i] = faceVal[randVal];
            faceVal[randVal] = itemp;
        }
    }
    return;
}
void selSort(string c[], int faceVal[], int NUMCARD) {
    for (int i = 0; i < NUMCARD-1; i++) {
        int minIndx = i;
        for (int j = i+1; j < NUMCARD;j++ ) {

            if (faceVal[j] > faceVal[minIndx]) {
                minIndx = j;
            }
            if (minIndx != i) {
                int temp = faceVal[minIndx];
                faceVal[minIndx] = faceVal[j];
                faceVal[j] = temp;

                string tempS = c[minIndx];
                c[minIndx] = c[j];
                c[j] = tempS;
            }
        }
    }
    return;
}

//-------------------------------------------------------------------------------------------------------------
-----------------------------
//End of array functions.
//-------------------------------------------------------------------------------------------------------------
-----------------------------

void game (int NUMCARD, int faceVal[], string c[],  vector<int> &deck, vector<string> &card,
int &bet,int &total, short &wins,int p1Hand, short dealer) {
    bool whoWon;

    getCard(c,faceVal,NUMCARD,p1Hand);
    getCard(c,faceVal,NUMCARD,p1Hand);
    cout << "Your hand is " << p1Hand << '\n';
    getCard(deck,card,dealer);
    getCard(deck,card,dealer);
```

```
if (dealer == 21 && p1Hand == 21) {
    cout << "Both people got blackjack. Push.\n";
    cout << "Your total is now " << betUpdt(bet,total,2) << '\n';
    return;
}
if (check21(dealer) == true) {
    cout << "Dealer got blackjack. \n";
    cout << "Your total is now " << betUpdt(bet,total,0) << '\n';
    return;
}
if (check21(p1Hand) == true) {
    cout << "Player got blackjack. \n";
    cout << "Your total is now " << betUpdt(bet,total,3) << '\n';
    wins++;
    return;
}
stndHit(NUMCARD,faceVal,c,p1Hand);
dealDrw(deck, card, dealer);
cout << "Dealer's hand is now " << dealer << '\n';
if (dealer == p1Hand) {
    cout << "Push.";
    cout << "Your total is now " << betUpdt(bet,total,2) << '\n';
    return;
}
whoWon = p1Hand>dealer?true:false;
if (p1Hand > 21) {
    cout << "You busted. You lost.\n";
    cout << "Your total is now " << betUpdt(bet,total,0) << '\n';
}
else if (dealer > 21) {
    cout << "Dealer busted. You won.\n";
    cout << "Your total is now " << betUpdt(bet,total,1) << '\n';
    wins++;
}
else {
    if (whoWon == true) {
        cout << "You won!\n";
        cout << "Your total is now " << betUpdt(bet,total,1) << '\n';
        wins++;
    }
```

```
        else {
            cout << "You lost!\n";
            cout << "Your total is now " << betUpdt(bet,total,0) << '\n';
        }
    }
    return;
}

void dealDrw(vector<int> &deck, vector<string> &card, short &dealer) {
    while (dealer < 17) {
        getCard(deck, card, dealer);
    }
}

void stndHit (int NUMCARD, int faceVal[], string c[], int &p1Hand) {
    unsigned short choice;
    while (p1Hand <= 21 && choice != 2){
        cout << "\nPress 1 to hit\nPress 2 to stand\n";
        cin>>choice;
        switch (choice) {
            case 1: getCard(c,faceVal,NUMCARD,p1Hand); cout <<"Your hand is now " << p1Hand
<< '\n';break;
            case 2: return;
        }
    }
    return;

}

bool check21 (short dealer) {
    if (dealer == 21) return true;
    else {
        return false;
    }
}

bool check21 (int p1Hand) {
    if (p1Hand == 21) return true;
    else {
        return false;
    }
}
//-----------------------------------------------------------------------------------------------------
//Anything below will be menu/betting functions
//-----------------------------------------------------------------------------------------------------

void menu(string c[], int faceVal[], int NUMCARD) {
    //menu for the beginning
    ifstream in;
    string welcome;
    unsigned short menu;
    int val;

    //opening welcome.txt which is just a notepad text with a "Welcome
    //To Christian's Blackjack Game!" and assigning it to string variable "welcome"
    in.open("welcome.txt");
    getline(in,welcome);
    in.close();

        //start of the menu (just menu in this do while loop)
    do {
            cout << welcome;
            cout << "\n1. Play Blackjack."<<
                    "\n2. How to play."<<
                    "\n3. Search deck for certain card."<<
                    "\n4. Quit game." << '\n';
            cin >> menu;
            //menu choice 2 which is the rules of the game
            if (menu == 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";
            }
            //menu choice 3 which is to quit
            if (menu == 4) {
                cout << "Goodbye!";
                exit(0);

            }
            if (menu == 3) {
                cout << "What card would you like to search for?\n"<<
                        "1 = Ace\n" <<
                        "10 = Jack\n" <<
                        "11 = Queen\n" <<
                        "12 = King\n";
                cin >> val;
                linSrch(c,faceVal,NUMCARD,val);
            }
            if (menu > 4) {
                cout << "Invalid option." << '\n';
            }
    }while (menu != 1);
    return;
}

void initBet(int &bet, int &total) {
    if (total == 0) {
        cout << "How much total are we planning on using to bet today?\n";
        cin >> total;
    }
    cout << "How much would you like to bet? Can't bet over $50,000.\n";
    cin>>bet;
    if (bet > 50000) {
        cout << "Your bet is now $50,000." << endl;
        bet = 50000;
    }

    total = total-bet;
    return;
}

int betUpdt (int bet, int &total, int update) {
    /* 0 = loss
     * 1 = win
     * 2 = push
```

```
  * 3 = blackjack
  */
 if (update == 0) {
    total = total;
    return total;
 }
 else if (update == 1) {
    total +=(bet*2);
    return total;
 }
 else if (update == 2) {
    total += bet;
    return total;
 }
 else if (update == 3) {
    total += (bet*2.5);
    return total;
 }
 return 0;
}


/*
 *  loop until every element in the array is checked to see if it matches val
 *
 * for ( i = 0; i < NUMBEROFCARDS; i++) {
 *     check if value is equal to value looking for
 *     cout << name of card << next line;
 *  }
 */
void linSrch (string c[], int faceVal[], int NUMCARD, int val) {
 for (int i = 0; i < NUMCARD; i++) {
    if (faceVal[i] == val) {
       cout << c[i] << '\n';
    }
 }
 return;
}
```