Project V2

Blackjack

CSC – 46090

Name: Anthony Vantino

Date: 07/30/15

**Introduction**

Blackjack

**Game Objective**

Blackjack is played against the casino so the main objective is to have a total hand value that exceeds the total hand value of the dealer without going over a total point value of 21.

A traditional 52 cards deck is used (or several decks) and each card has a point value attached to it. Aces count as 1 or 11, Face cards (King, Queen and Jack) count as 10 and all other cards keep the value that is printed on them.

**How to Play**

The dealer deals each player two cards facing up. The dealer also gets two cards but only one is facing up and the other is facing down (known as the hole card). Players are allowed to draw additional cards to the total hand value of 21.

Once a player drew a card that takes his total hand value above 21 they bust out of the game. Either the dealer or player can win the hand with low valued hands in situations when one or the other busted out of the game.

**Summary**

Project size: 683 Lines

Number of Variables: 12

The Number of methods: 10

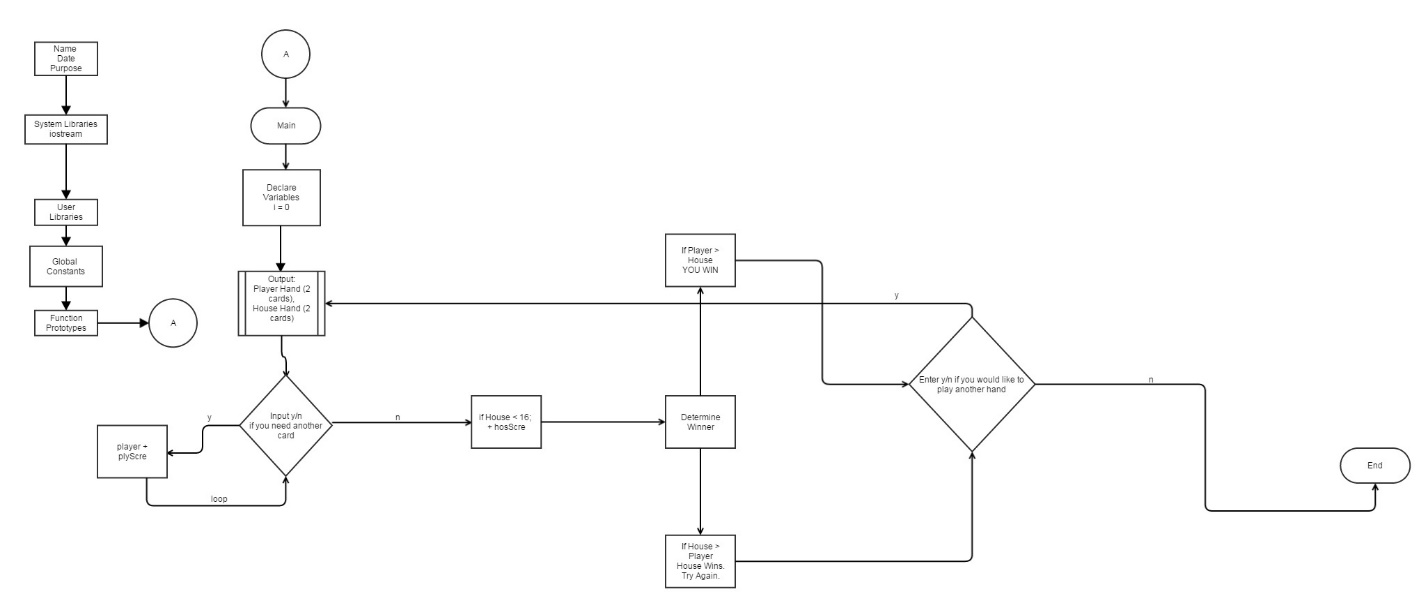
This project includes many of the concepts that we learned throughout the Summer Semester. I used Gaddis 8th Edition for reference points. I’ve expanded this project greatly since first written but I feel there are still things that could be added to a final product.

It took around 2-3 weeks. Once I got comfortable with arrays I knew I could expand the deck features a lot more and have a more realistic and immersive playing experience. Once I expanded the deck I went through and ironed out a lot of the overall mess that was version 1. It has a much cleaner final output and drawing sequence.

**Description**

The main point that I programmed this project is dealing, and how the cards are dealt to the player and the House as the game progresses.

**Flowchart**

**Pseudo Code**

*Initialize*

*If the start button is pressed*

*Display greeting message that the round as begun*

*Display player hand and the house hand*

*If player needs another card*

*Get and output another card*

*Else*

*Begin House turn*

*Output cards until house is greater than 16*

*If Player Hand is greater than House Hand*

*Output Congratulations string*

*Else*

*Output Try Again string*

*If player inputs to play again*

*Loop*

*Else*

*Exit Program*

**Major Variables**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| **Type** | **Variable Name** | | **Description** | |  | **Location** |
| **integer** | **I,j** |  | **counter** |  |  | **main** |
|  |  |  |  |  |  |  |
|  | **deck** |  | **deck array** | |  | **playHnd** |
|  |  |  |  |  |  | **shuffle** |
|  |  |  |  |  |  | **showCrd** |
|  |  |  |  |  |  | **getTop** |
|  |  |  |  |  |  | **hitStnd** |
|  | **size** |  | **size of array** | |  | **shuffle** |
|  | **card** |  | **card value** | |  | **showCrd** |
|  |  |  |  |  |  | **cardVal** |
|  | **numCrds** |  | **Number of Cards in play** | | | **showCrd** |
|  | **low** |  | **low of rand** | |  | **rndNum** |
|  | **high** |  | **high of rand** | |  | **rndNum** |
|  | **hand** |  | **hand array** | |  | **add** |
|  | **cardAdd** |  | **Add card to hand value** | | | **add** |
|  | **dealHnd** |  | **dealer hand** | |  | **hitStnd** |
|  |  |  |  |  |  | **winner** |
|  |  |  |  |  |  | **bust** |
|  |  |  |  |  |  | **blckJck** |
|  |  |  |  |  |  | **sftHrd** |
|  |  |  |  |  |  | **sftHrdD** |
|  |  |  |  |  |  | **score** |
|  | **playHnd** |  | **player hand** | |  | **hitStnd** |
|  |  |  |  |  |  | **winner** |
|  |  |  |  |  |  | **bust** |
|  |  |  |  |  |  | **blckJck** |
|  |  |  |  |  |  | **score** |
|  |  |  |  |  |  | **sftHrd** |
|  |  |  |  |  |  | **sftHrdD** |
| **char** | **play** |  | **to play another or quit** | | | **playHnd** |
|  |  |  |  |  |  | **playAgn** |
|  |  |  |  |  |  | **bust** |
|  |  |  |  |  |  | **play** |
|  | **draw** |  | **to draw another or quit** | | | **playDrw** |

**C++ Constructs**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Chapter** | **New Syntax and keywords** | | |  | **Location** |  |
| **2** | **#include** |  |  |  | **system libraries** | |
|  | **namespace std** | |  |  | **system libraries** | |
|  | **main()** |  |  |  | **main** |  |
|  | **int** |  |  |  | **function prototypes** | |
|  |  |  |  |  | **main** |  |
|  |  |  |  |  | **blckjck function** | |
|  |  |  |  |  | **winner function** | |
|  |  |  |  |  | **deal function** | |
|  |  |  |  |  | **houseHt function** | |
|  |  |  |  |  | **hit function** | |
|  |  |  |  |  | **random function** | |
|  | **char** |  |  |  | **hit function** | |
|  |  |  |  |  | **main** |  |
|  | **return** |  |  |  | **main** |  |
|  |  |  |  |  | **deal function** | |
|  |  |  |  |  | **random function** | |
| **4** | **if** |  |  |  | **hit function** | |
|  |  |  |  |  | **houseHt function** | |
|  |  |  |  |  | **winner function** | |
| **4** | **else** |  |  |  | **winner function** | |
|  |  |  |  |  | **hit function** | |
|  |  |  |  |  | **deal function** | |
|  |  |  |  |  | **main** |  |
| **5** | **while** |  |  |  | **main** |  |
|  |  |  |  |  | **deal function** | |
|  |  |  |  |  | **hit function** | |
| **5** | **for** |  |  |  | **deal function** | |
| **6** | **void** |  |  |  | **function prototypes** | |
|  |  |  |  |  | **blckjck function** | |
|  |  |  |  |  | **winner function** | |
|  |  |  |  |  | **hit function** | |
|  |  |  |  |  | **houseHt function** | |

**Reference**

1. Gaddis 8th Edition
2. [www.cplusplus.com](http://www.cplusplus.com) (forum for helping people with C++ coding)

**Program**

//System Libraries

#include <iostream>

#include <cstdlib>

#include <limits>

#include <fstream>

#include <string>

using namespace std;

//User Libraries

//Global Constants

//Function Prototypes

void playHnd();

void crtDeck(int deck[]);

void shuffle(int deck[], int size);

void showCrd(int card);

void showCrd(const int deck[], int numCrds, bool hide);

int rndNum(int low, int high);

int cardVal(int card);

int getTop(int deck[]);

void add(int hand[], int cardAdd);

void hitStnd(int dealHnd[], int deck[], int playHnd[]);

int getHand(const int hand[]);

bool playDrw(char&);

bool playAgn(char&);

void winner(const int playHnd[], const int dealHnd[]);

void bust(const int playHnd[], const int dealHnd[], char play);

void blckJck(const int playHnd[], const int dealHnd[], char play);

void score(const int playHnd[], const int dealHnd[]);

void sftHrd(int playHnd[]);

void sftHrdD(int dealHnd[], int playHnd[]);

//Execution Begins here!

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

string txt[1000];

string line;

int i=0;

ifstream rules ("rules.txt");

if (rules.is\_open()){

while (!rules.eof()){

getline(rules,line);

txt[i]=line;

i++;

}

}

rules.close();

for (int j=0; txt[j]!="\0"; j++)

cout << txt[j] << endl;

//Play BlackJack

playHnd();

//Winter is Coming!

return 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Play Hand \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Play a hand of Blackjack (Main Game Logic)

\* inputs:

\* hit -> Draw another card

\* stand -> End your turn

\* playAgn -> Play another hand of Blackjack

\* Outputs:

\* Winner -> winner of the hand

\* Next hand -> The next hand of Blackjack

\*/

void playHnd(){

char play = 'n';

do{

char draw = 'h';

//Initialize and Shuffle Deck

int deck[52];

crtDeck(deck);

shuffle(deck,51);

//Declare Player Hands

int playHnd[10] = {0};

int dealHnd[10] = {0};

//Deal

add(playHnd, getTop(deck));

add(dealHnd, getTop(deck));

add(playHnd, getTop(deck));

add(dealHnd, getTop(deck));

cout<<"///////////////////////////////////////////////////////\n"<<endl;

cout<<"The Dealer shuffled and dealt you each your hands.\n"<<endl;

//Check for Auto Win

blckJck(playHnd, dealHnd, 'Y');

//Display Hands

cout<<"Your hand ";

showCrd(playHnd, 10, false);

cout<<"\nDealer's hand ";

showCrd(dealHnd, 10, true);

//Soft/Hard Ace if applicable

sftHrd(playHnd);

//Hit

while(playDrw(draw)){

//Deal Player

add(playHnd, getTop(deck));

cout<<"The Dealer dealt you another card."<<endl;

//Display new hand

cout<<"Your hand "<<endl;

showCrd(playHnd, 10, false);

//Soft/Hard Ace if applicable

sftHrd(playHnd);

//Check bust

bust(playHnd, dealHnd, 'y');

//Check Winner

blckJck(playHnd, dealHnd, 'Y');

}

//Dealer hits until hand >= 16

hitStnd(dealHnd, deck, playHnd);

//Check bust

bust(playHnd, dealHnd, 'y');

//Check Winner

blckJck(playHnd, dealHnd, 'Y');

//Compare Hands & Determine Winner

winner(playHnd, dealHnd);

}

while(playAgn(play));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Create Deck \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Create the deck used for blackjack

\* inputs:

\*

\* Outputs:

\* deck -> 52 cards

\*/

void crtDeck(int deck[]){

int rank = 101;

int i = 0;

//Hearts

for(i = 0; i <= 13; i++){

deck[i] = rank++;

}

//Diamonds

for(i = 13; i <= 26; i++){

deck[i] = rank++ + 100 - 14;

}

//Clubs

for(i = 26; i <= 39; i++){

deck[i] = rank++ + 200 - 28;

}

//Spades

for(i = 39; i <= 51; i++){

deck[i] = rank++ + 300- 42;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Shuffle \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Shuffle the deck after every hand

\* inputs:

\*

\* Outputs:

\* new deck -> a new deck with the same cards

\*/

void shuffle(int deck[], int size){

for(int i = 0; i < 500; i++){

int t1 = 0;

int r1 = rndNum(0,size);

int r2 = rndNum(0,size);

//Clone first card, replace first with new card,

//replace new card with clone

t1 = deck[r1];

deck[r1] = deck[r2];

deck[r2] = t1;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Show Card \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Display what the card is when in play

\* inputs:

\*

\* Outputs:

\* card -> Suit and rank

\*/

void showCrd(int card){

if(card == 0){

cout<<"";

}

//Ranks

else{

switch(card%100){

case 1:

cout<<"A";

break;

case 11:

cout<<"J";

break;

case 12:

cout<<"K";

break;

case 13:

cout<<"Q";

break;

case 14:

cout<<"A";

break;

default:

cout<<card%100;

}

}

//Show nothing for non card (ie.0)

if(card == 0){

cout<<"";

}

//Suits

else{

//Hearts

if((card >= 101) && (card <= 114)){

cout<<static\_cast<char>(3);

}

//Diamonds

else if((card >= 201) && (card <= 314)){

cout<<static\_cast<char>(5);

}

//Clubs

else if((card >= 301) && (card <= 314)){

cout<<static\_cast<char>(5);

}

//Spades

else if((card >= 401) && (card <= 414)){

cout<<static\_cast<char>(6);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Show Card \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Show the cards dealt in the hand

\* inputs:

\*

\* Outputs:

\* hide -> dealer's first card (just like in real life)

\* Display -> The card

\*/

void showCrd(const int deck[], int numCrds, bool hide){

//Hide Dealer's first card

if(hide){

cout<<"\*\*";

}

//Show Dealer's first card

else{

showCrd(deck[0]);

cout<<" ";

}

//Display all the cards in hand

for(int i = 1; i < numCrds; i++){

//Show Cards

if(deck[i]!=0){

showCrd(deck[i]);

cout<<" ";

}

//Show nothing for non cards(ie. 0)

else{

cout<<"";

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Random Number \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Give player a random number

\* inputs:

\*

\* Outputs:

\* randNum -> Random Number

\*/

int rndNum(int low, int high){

static bool first = true;

int randNum;

//If called, seed random number generator

if(first){

srand(static\_cast<unsigned int>(time(0)));

first=false;

}

//Generate Random Number

randNum = rand()%(high-low+1) + low;

return randNum;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Card Value \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Play a hand of Blackjack

\* inputs:

\*

\* Outputs:

\* card -> Card value

\*/

int cardVal(int card){

int cardVal;

//Get card value

switch(card%100){

case 1:

cardVal = 11;

break;

case 11:

case 12:

case 13:

cardVal = 10;

break;

case 14:

cardVal = 1;

break;

default:

cardVal = (card%100);

}

return cardVal;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Get Top \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Get top card of the deck

\* inputs:

\*

\* Outputs:

\* topCard -> the top card in the deck of 52

\*/

int getTop(int deck[]){

int topCard = 0;

//Loop through deck

for(int i = 0; i < 51; i++){

//Find a card

if(deck[i]!=0){

//Clone card, replace with empty value, return clone

topCard = deck[i];

deck[i] = 0;

return topCard;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Add \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Add the cards together for a total

\* inputs:

\*

\* Outputs:

\* hand -> your final total at the end of the hand

\*/

void add(int hand[], int cardAdd){

//Loop through hand

for(int i = 0; i < 9; i++){

//If empty add card

if(hand[i] == 0){

hand[i] = cardAdd;

break;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Hit Stand \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Make the dealer hit until he is at or above 16

\* inputs:

\*

\* Outputs:

\* dealHnd -> once 16 dealer stops drawing cards

\*/

void hitStnd(int dealHnd[], int deck[], int playHnd[]){

//Loop through Dealer hand

for(int i = 0; i < 9; i++){

//If dealer must hit

if(getHand(dealHnd) < 16){

add(dealHnd, getTop(deck));

//Ensure Aces used intelligently

sftHrdD(dealHnd, playHnd);

}

//Display how many cards dealer hit

else{

if(i == 0){

cout<<"The Dealer stands."<<endl;

break;

}

else if(i == 1){

cout<<"The Dealer hit and stands"<<endl;

break;

}

else{

cout<<"The Dealer hit "<<i<<" cards and stands"<<endl;

break;

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Get Hand \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Get Player/Dealer hand

\* inputs:

\*

\* Outputs:

\* addCrd -> add card to player/dealer hand

\*/

int getHand(const int hand[]){

int addCrd = 0;

//Loop through hand and add cards

for(int i = 0; i < 9; i++){

addCrd = addCrd + cardVal(hand[i]);

}

return addCrd;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Play Again \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Play another hand of Blackjack

\* inputs:

\* play -> Yes or No

\* Outputs:

\* True -> play again

\* False -> End game

\*/

bool playAgn(char& play){

//Prompt user to play again

cout<<"\n\nWould you like to play another hand? (y/n) ";

cin>>play;

cout<<"\n\n";

//Goto main if yes

if(play == 'y' || play == 'Y'){

playHnd();

return(true);

}

//else Exit

else{

return(false);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Player Draw \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Ask player if they would like to hit

\* inputs:

\* draw -> yes or no

\* Outputs:

\* true -> give player a card

\* false -> End players turn begin Dealer turn

\*/

bool playDrw(char& draw){

//Prompt user to see if they would like to hit/stand

cout<<"\n\nWould you like to hit or stand? (h/s) ";

cin>>draw;

cout<<"\n";

//If yes start loop to draw another card

if(draw == 'h' || draw == 'H'){

return(true);

}

//else skip

else{

return(false);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Winner \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Calculate the winner of the round

\* inputs:

\*

\* Outputs:

\* playScr -> Player is the winner of the hand

\* dealScr -> Dealer is the winner of the hand

\*/

void winner(const int playHnd[], const int dealHnd[]){

int playScr = getHand(playHnd);

int dealScr = getHand(dealHnd);

//Display Scoreboard

score(playHnd, dealHnd);

//If player scores less than 22

if((playScr < 22) && (playScr > dealScr) || (dealScr > 21) && (playScr < 22)){

cout<<"\nYou Win!"<<endl;

}

//Lose loop

else{

//Tie

if(playScr == dealScr){

cout<<"\nPush, no winner."<<endl;

}

//Lose

else{

cout<<"\nYou Lose."<<endl;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Bust \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: To find out if player or dealer busted

\* inputs:

\*

\* Outputs:

\* playAgn -> Would you like to play again Yes or No

\*/

void bust(const int playHnd[], const int dealHnd[], char play){

int playScr = getHand(playHnd);

int dealScr = getHand(dealHnd);

//Check play bust

if(playScr > 21){

cout<<"You bust with "<<getHand(playHnd)<<" point."<<endl<<endl;

playAgn(play);

}

//Check Dealer bust

else if(dealScr > 21){

score(playHnd, dealHnd);

cout<<"\n\nThe Dealer busted. You Win!"<<endl<<endl;

playAgn(play);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Blackjack \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Determine if the dealer or player has a perfect 21

\* inputs:

\*

\* Outputs:

\* playAgn -> When winner is determined ask if they would like to play again

\*/

void blckJck(const int playHnd[], const int dealHnd[], char play){

int playScr = getHand(playHnd);

int dealScr = getHand(dealHnd);

//If play has blackjack

if((playScr == 21) && (dealScr != 21)){

cout<<"\n\n";

score(playHnd, dealHnd);

cout<<"\n21! You got Blackjack! You Win!"<<endl;

playAgn(play);

}

//If both have blackjack

else if((playScr == 21) && (dealScr == 21)){

score(playHnd, dealHnd);

cout<<"\nThe Dealer and you both got Blackjack. Push."<<endl;

playAgn(play);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Score \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Play a hand of Blackjack

\* inputs:

\*

\* Outputs:

\* playHnd -> Final score at end of round

\* dealHnd -> Final score at end of round

\*/

void score(const int playHnd[], const int dealHnd[]){

//Display Hand

cout<<"Player Hand ";

showCrd(playHnd, 10, false);

cout<<" ("<<getHand(playHnd)<<"pts)."<<endl;

cout<<"vs"<<endl;

//Display Dealer hand

cout<<"Dealer hand ";

showCrd(dealHnd, 10, false);

cout<<" ("<<getHand(dealHnd)<<"pts)."<<endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Soft Hard \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Ask player if they would like their Ace to be a 1 or 11

\* inputs:

\* sftHrd -> 1 or 11

\* Outputs:

\* ace -> either 1 or 11 value

\*/

void sftHrd(int playHnd[]){

//Check hand

for(int i = 0; i < 9; i++){

int ace = cardVal(playHnd[i]);

int sftHrd;

//Prompt User input

if(ace == 1 || ace == 11){

cout<<"\n\nWould you like your Ace to count as 1 or 11?"<<endl;

cin>>sftHrd;

//If 1

if(sftHrd == 1){

if(ace == 1){

playHnd[i] = playHnd[i] - 13;

}

}

else if(sftHrd != 1 || sftHrd != 11){

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(),'\n');

//Display Error

cout<<"\nPlease enter the number 1 or 11."<<endl;

return;

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Soft Hard Dealer \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Purpose: Determine if the dealers Ace should be an 1 or 11

\* inputs:

\*

\* Outputs:

\* dealHnd -> Based on if the 11 or 1 would be the better choice

\*/

void sftHrdD(int dealHnd[], int playHnd[]){

//Check hand

for(int i = 0; i < 9; i++){

//Find Ace

if(cardVal(dealHnd[i]) == 1 || cardVal(dealHnd[i]) == 11){

//If ace 11

if(cardVal(dealHnd[i]) ==11){

//Change to 1 if it beats player

if(getHand(dealHnd) - cardVal(dealHnd[i]) + 1 > getHand(playHnd)){

//Not bust

if(getHand(dealHnd) - cardVal(dealHnd[i]) + 1 < 22){

dealHnd[i] = dealHnd[i] + 13;

}

}

//Else if hand bust make 1

else if(getHand(dealHnd) > 21){

dealHnd[i] = dealHnd[i] + 13;

}

}

//Else ace is 1

else{

//Change to 11 if it beats player

if(getHand(dealHnd) - cardVal(dealHnd[i]) + 11 > getHand(playHnd)){

//not bust

if(getHand(dealHnd) - cardVal(dealHnd[i]) + 11 < 22){

dealHnd[i] = dealHnd[i] - 13;

}

}

}

}

}

}