Project 2
<Blackjack Game>

CSC-5 42644
Name: Gutierrez, Armando
Date: 06/03/17

Introduction

Title: Blackjack

Blackjack is a card game in which one or more players attempt to get as close to the sum value of 21 without getting more. A value of 22 or higher is an automatic "bust" which means a loss. Each player is simply trying to beat the dealer to 21 and is independent of the other players. The Ace card's value is 1 or 11, based on which is more advantageous to the user. Cards 2-10 retain their face value. King, Queen, and Jack are valued at 10. The player(s) and the dealer each receive two cards, one of which is face down. It is common strategy to "hit", or receive another card from the deck, if the sum of the first two cards is less than 17. Once the player no longer wants to hit, they may choose to "stay", which means they no longer receive cards in hopes that their sum is higher than the dealer's. The player also has the option of "doubling down" after receiving his or her first two cards. Doubling down simply means that the player will receive their third and final card, while doubling the amount of money they have bet. The dealer hits until a sum of 17 or higher is reached. Receiving a sum of 21 results in a "blackjack" and counts as a win. If no one has busted or gotten a blackjack, the winner is the one whose sum is closest to 21. The main difference between my game and regular blackjack how it is usually played is that each player plays against the dealer separately rather than the dealer playing everybody else.

Summary

Project size: about 400 lines

The number of variables: about 25

This project includes all the concepts that we have learned from the two books by Tony Gaddis and Walter Savitch. It also contains the code used from the previous project. The project took approximately 35 hours to finish from start to finish as I had to apply creativity to get the game to work the way I had envisioned. Because of this, it was a very beneficial experience. Not only did it solidify the concepts in my mind, but it also gave me an insight into the mindset of an engineer.

Description

The main aspect of this program is the use of the random number generator, and how to update the sum of these numbers using single and two dimensional arrays over several games or repetitions of the program.

Important Note: I was having trouble with NetBeans on mac and needed to include a "mac fix" at the beginning of my program. For some reason my 2-D array would be affected if it the fix were not there.

Cross Reference Check-Off List

Chapter	Section	Topic	Line Number in Code(if not obvious!)
2	2	cout	everywhere
	3	libraries	Iostream, iomanip, cmath, cstdlib, fstream, string, ctime
	4	variables/literals	50-63
	5	Identifiers	66
	6	Integers	50-63
	7	Characters	53,58
	8	Strings	66-67
	9	Floats. No doubles	228
	10	Bools	144, 367
	12	Variables 7 characters or less	All of them
	14	Arithmetic Operations	All throughout
	15	Comments 20%+	All throughout
	16	Named Constants	31-33
3	1	cin	All throughout
	2	Math Expression	232
	5	Type Casting	228,230
	7	Formatting Output	223
	8	Strings	66-67
	9	Math Library	20

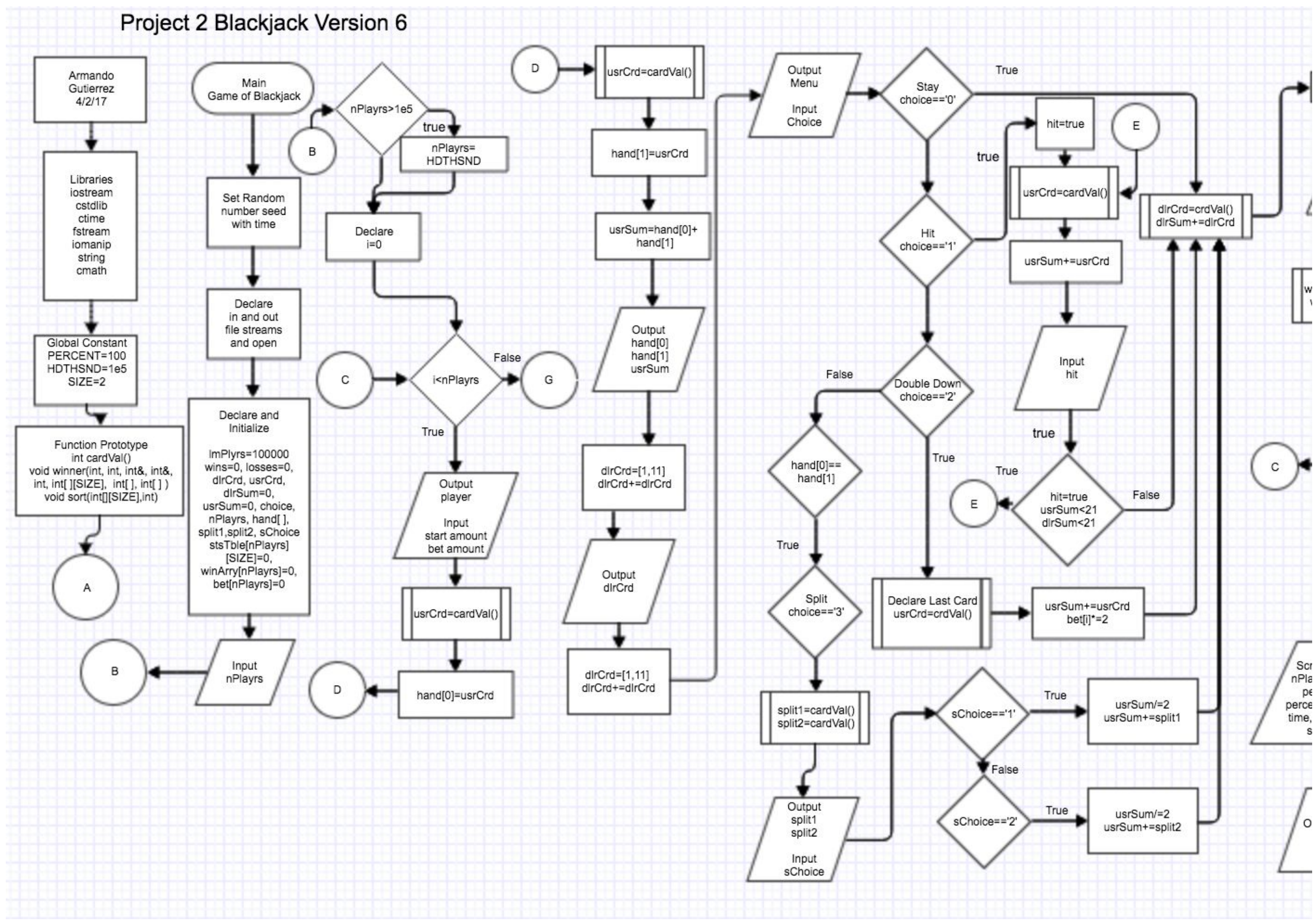
4	1	Relational Operators	320,334, All throughout
	2	if	317-360
	4	If-else	317-360
	5	Nesting	317-360
	6	If-else-if	317-360
	8	Logical Operators	341
	11	Validating user input	140
	13	Conditional Operator	59
	14	Switch	142
5	1	Increment/Decrement	322-324
	2	While	71
	5	Do-while	197
	6	For loop	All throughout
	11	Files input/output both	69
6	3	Function Prototypes	36
	5	Passing by value	37
	8	Returning values from functions	408
	10	No Global variables	30
	11	Static Local	95,220
	13	Reference Parameters	37,314

7	4	Array Initialization	74-84
	6	Processing Arrays	All throughout
	7	Parallel Arrays	235,270
	8	Arrays as function arguments	37-39
	9	2-D Arrays	59
8	1	Binary Search	387
	3	Bubble Sort	365

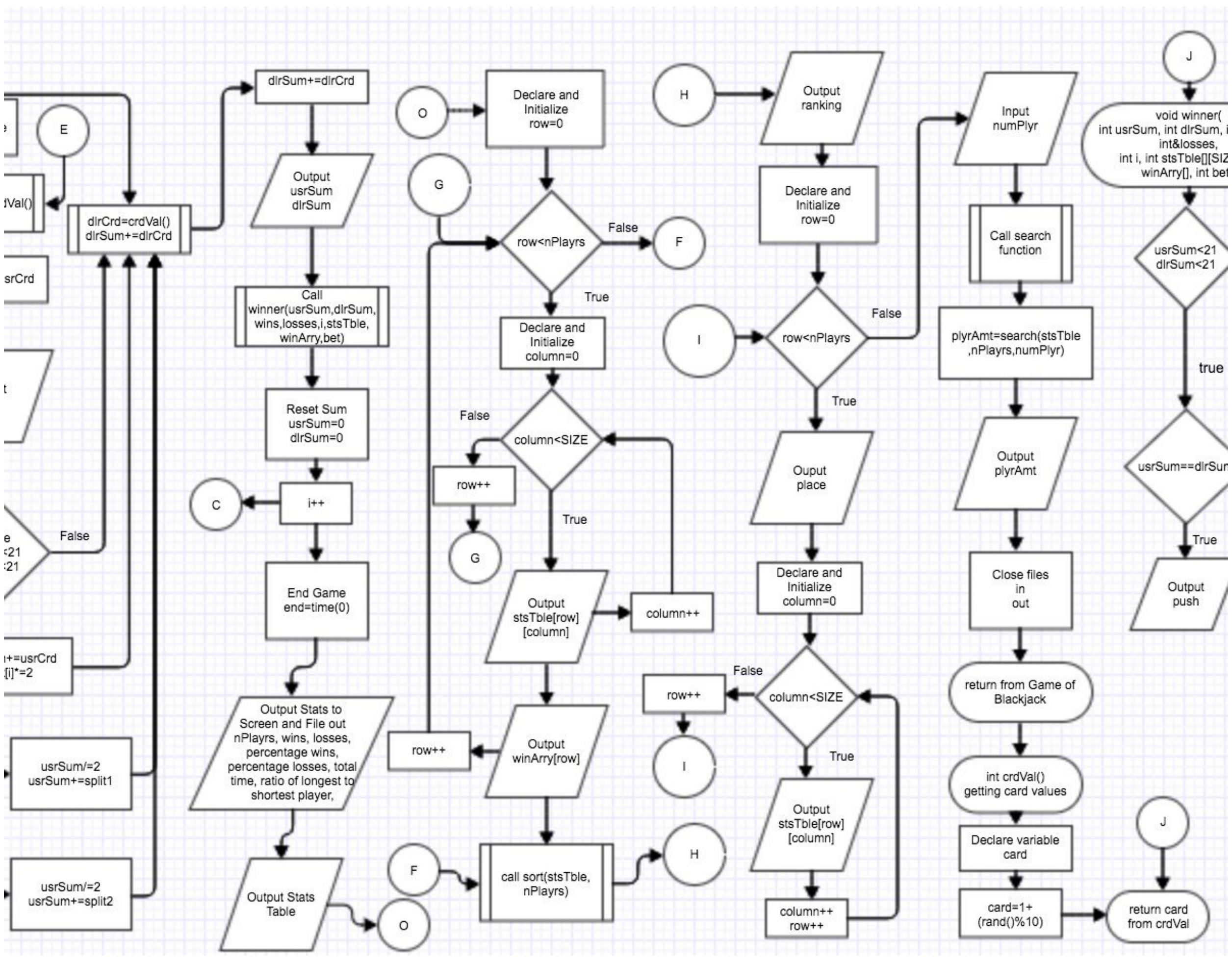
Flow Chart

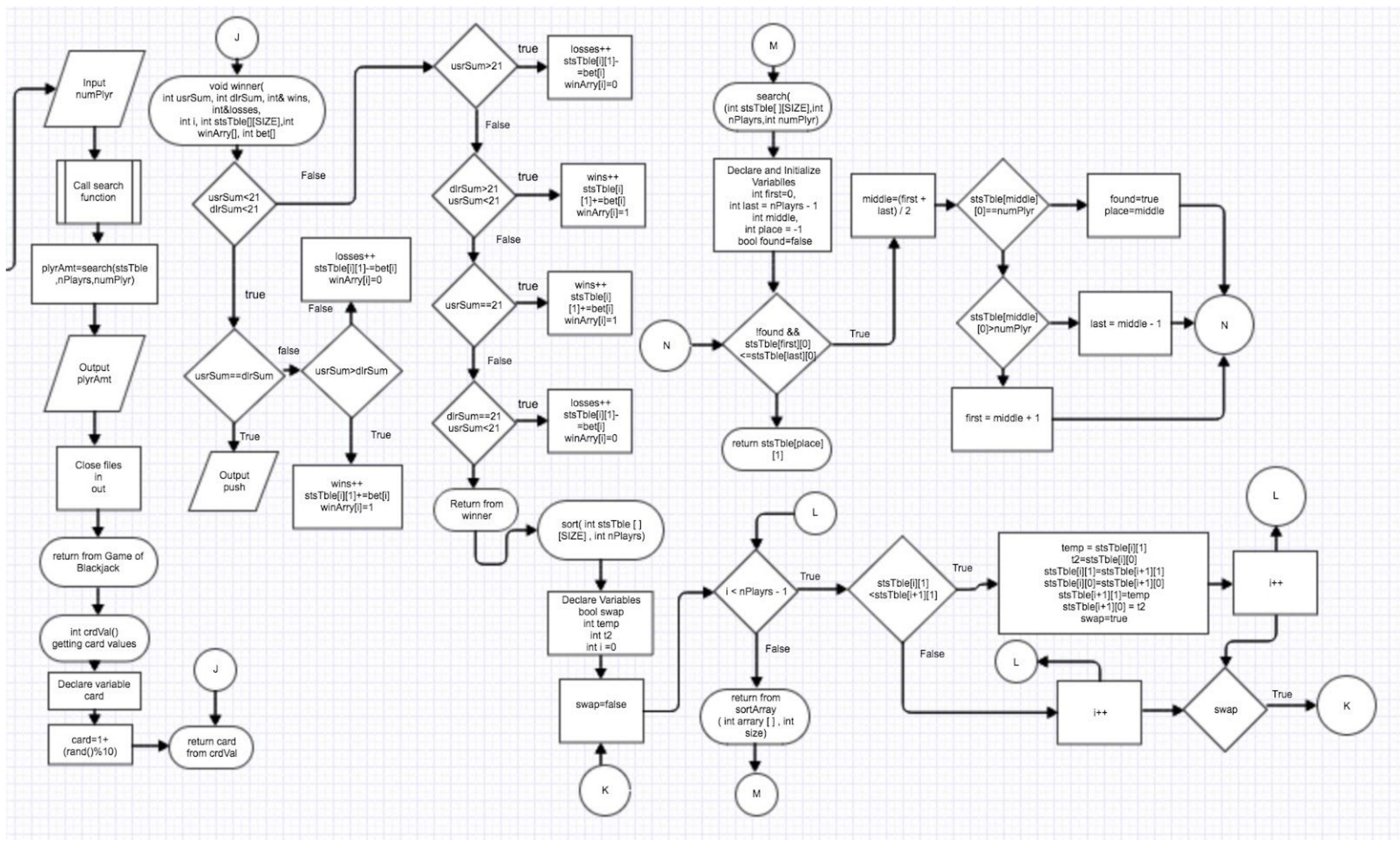
Due to the large size of the flowchart, I will paste segments of it. For full flowchart visit: https://www.gliffy.com/go/publish/11941059

Important note about the link!: For some reason the words in the the flowchart are not contained in the shapes they are written in. I attempted to fix this problem multiple times, but it seems gliffy's viewer link won't co-oporate. If it is too much of a problem, email me at armandogutierrez4802@gmail.com, and I can send you my gliffy.com login information.



Project 2 Blackjack Version 6
Armando Gutierrez 4/2/17
Libraries iostream cstdlib ctime fstream iomanip string cmath
Global Constant PERCENT=100 HDTHSND=1e5 SIZE=2
Function Prototype int cardVal() void winner(int, int, int&, int&, int, int[ ][SIZE], int[ ], int[ ]) void sort(int[][SIZE],int)
A
Main Game of Blackjack
Set Random number seed with time
Declare in and out file streams and open
Declare and Initialize
lmPlyrs=100000 wins=0, losses=0, dlrCrd, usrCrd, dlrSum=0, usrSum=0, choice, nPlayrs, hand[ ], split1,split2, sChoice stsTble[nPlayrs][SIZE]=0, winArry[nPlayrs]=0, bet[nPlayrs]=0
Input nPlayrs
B
nPlayrs>1e5
true
nPlayrs= HDTHSND
Declare i=0
C
i<nPlayrs
False
G
True
Output player
Input start amount bet amount
usrCrd=cardVal()
hand[0]=usrCrd
D
usrCrd=cardVal()
hand[1]=usrCrd
usrSum=hand[0]+ hand[1]
Output hand[0] hand[1] usrSum
dlrCrd=[1,11] dlrCrd+=dlrCrd
Output dlrCrd
dlrCrd=[1,11] dlrCrd+=dlrCrd
Output Menu
Input Choice
Stay choice=='0'
True
Hit choice=='1'
true
hit=true
E
usrCrd=cardVal()
usrSum+=usrCrd
Input hit
true
hit=true usrSum<21 dlrSum<21
False
dlrCrd=crdVal() dlrSum+=dlrCrd
Double Down choice=='2'
False
True
hand[0]== hand[1]
Split choice=='3'
split1=cardVal() split2=cardVal()
Output split1 split2
Input sChoice
Declare Last Card usrCrd=crdVal()
usrSum+=usrCrd bet[i]*=2
sChoice=='1'
usrSum/=2 usrSum+=split1
sChoice=='2'
usrSum/=2 usrSum+=split2


E
dVal()
srCrd
False
+=usrCrd
[i]*=2
usrSum/=2
usrSum+=split1
usrSum/=2
usrSum+=split2
dlrCrd=crdVal()
dlrSum+=dlrCrd
dlrSum+=dlrCrd
Output
usrSum
dlrSum
Call
winner(usrSum,dlrSum,
wins,losses,i,stsTble,
winArry,bet)
Reset Sum
usrSum=0
dlrSum=0
C
i++
End Game
end=time(0)
Output Stats to
Screen and File out
nPlayrs, wins, losses,
percentage wins,
percentage losses, total
time, ratio of longest to
shortest player,
Output Stats
Table
O
O
Declare and
Initialize
row=0
G
row<nPlayrs
False
F
True
Declare and
Initialize
column=0
False
column<SIZE
row++
G
True
Output
stsTble[row]
[column]
column++
row++
Output
winArry[row]
F
call sort(stsTble,
nPlayrs)
H
H
Output
ranking
Declare and
Initialize
row=0
I
row<nPlayrs
False
True
Ouput
place
Declare and
Initialize
column=0
False
row++
column<SIZE
I
True
Output
stsTble[row]
[column]
column++
row++
Input
numPlyr
Call search
function
plyrAmt=search(stsTble
,nPlayrs,numPlyr)
Output
plyrAmt
Close files
in
out
return from Game of
Blackjack
int crdVal()
getting card values
Declare variable
card
card=1+
(rand()%10)
J
return card
from crdVal
J
void winner(
int usrSum, int dlrSum, i
int&losses,
int i, int stsTble[][SIZ
winArry[], int bet
usrSum<21
dlrSum<21
true
usrSum==dlrSum
True
Output
push


Input numPlyr
Call search function
plyrAmt=search(stsTble ,nPlayrs,numPlyr)
Output plyrAmt
Close files in out
return from Game of Blackjack
int crdVal() getting card values
Declare variable card
card=1+ (rand()%10)
return card from crdVal
J
void winner(int usrSum, int dlrSum, int& wins, int&losses, int i, int stsTble[][SIZE],int winArry[], int bet[]
usrSum<21 dlrSum<21
False
true
usrSum==dlrSum
false
True
Output push
usrSum>dlrSum
False
losses++ stsTble[i][1]-=bet[i] winArry[i]=0
True
wins++ stsTble[i][1]+=bet[i] winArry[i]=1
usrSum>21
true
losses++ stsTble[i][1]-=bet[i] winArry[i]=0
False
dlrSum>21 usrSum<21
true
wins++ stsTble[i][1]+=bet[i] winArry[i]=1
False
usrSum==21
true
wins++ stsTble[i][1]+=bet[i] winArry[i]=1
False
dlrSum==21 usrSum<21
true
losses++ stsTble[i][1]-=bet[i] winArry[i]=0
Return from winner
M
search((int stsTble[ ][SIZE],int nPlayrs,int numPlyr)
Declare and Initialize Variablles int first=0, int last = nPlayrs - 1 int middle, int place = -1 bool found=false
N
!found && stsTble[first][0] <=stsTble[last][0]
True
return stsTble[place][1]
middle=(first + last) / 2
stsTble[middle][0]==numPlyr
found=true place=middle
stsTble[middle][0]>numPlyr
last = middle - 1
first = middle + 1
N
sort(int stsTble [ ][SIZE] , int nPlayrs)
Declare Variables bool swap int temp int t2 int i =0
swap=false
K
L
i < nPlayrs - 1
True
False
return from sortArray (int arrary [ ] , int size)
M
stsTble[i][1] <stsTble[i+1][1]
True
False
temp = stsTble[i][1] t2=stsTble[i][0] stsTble[i][1]=stsTble[i+1][1] stsTble[i][0]=stsTble[i+1][0] stsTble[i+1][1]=temp stsTble[i+1][0] = t2 swap=true
i++
L
L
i++
swap
True
K

Pseudocode

```
/*
 * File:   main.cpp
 * Author: Armando Gutierrez
 * Created on May 17, 2017, 9:28 PM
 */

//System Libraries
                              //Input - Output Library
                              //Time for rand
                              //Srand to set the seed
                              //Format the output
                              //File I/O
                              //Strings
                              //Math functions

//User Libraries
```

```
//Global Constants
                   //Conversion to Percent
                 //One Hundred Thousand
            //Number of columns for 2-D array

//Function Prototypes
        //Returns new card value
                                   //Output winner
                 //Bubble sort
                   //Binary search

//Executable code starts here


  //Set the random number seed

  //Declare file and game variables
            //Input File
             //Output File
               //Output File
              //Dealer's/User's card
                 //Dealer's/User's sum of cards
            //User's choice
                 //Number of wins/losses
                       //Number of games/Limit to games
              //Cards in first hand
             //The third cards in each hand for split option
               //Split choice
                       //Statistics Table
                 //Initial amount of money
              //Bets
            //Number of player
          //Player's amount

  //Initialize variables
                          //String name
                             //String name

                 //Open the Input file
                 //Open the Output file
                //Last value in file becomes the number of games
```

```
                                              //Limit games
                                        //Initialize 2-D array

                                        //Initialize money array

                                  //Initialize bet array

//Mac fix

//Start the game


//Enter starting amount and bet amount



  //User first card


  //User's second card


  //Update and display user's sum


  //Show dealer's card 1

  //Update dealer's sum


  //Dealer's second card. Do not display


  //Update dealer's sum


  //Output menu



           //Enter user's choice to hit or stay

             //Switch statement with 3 cases
```

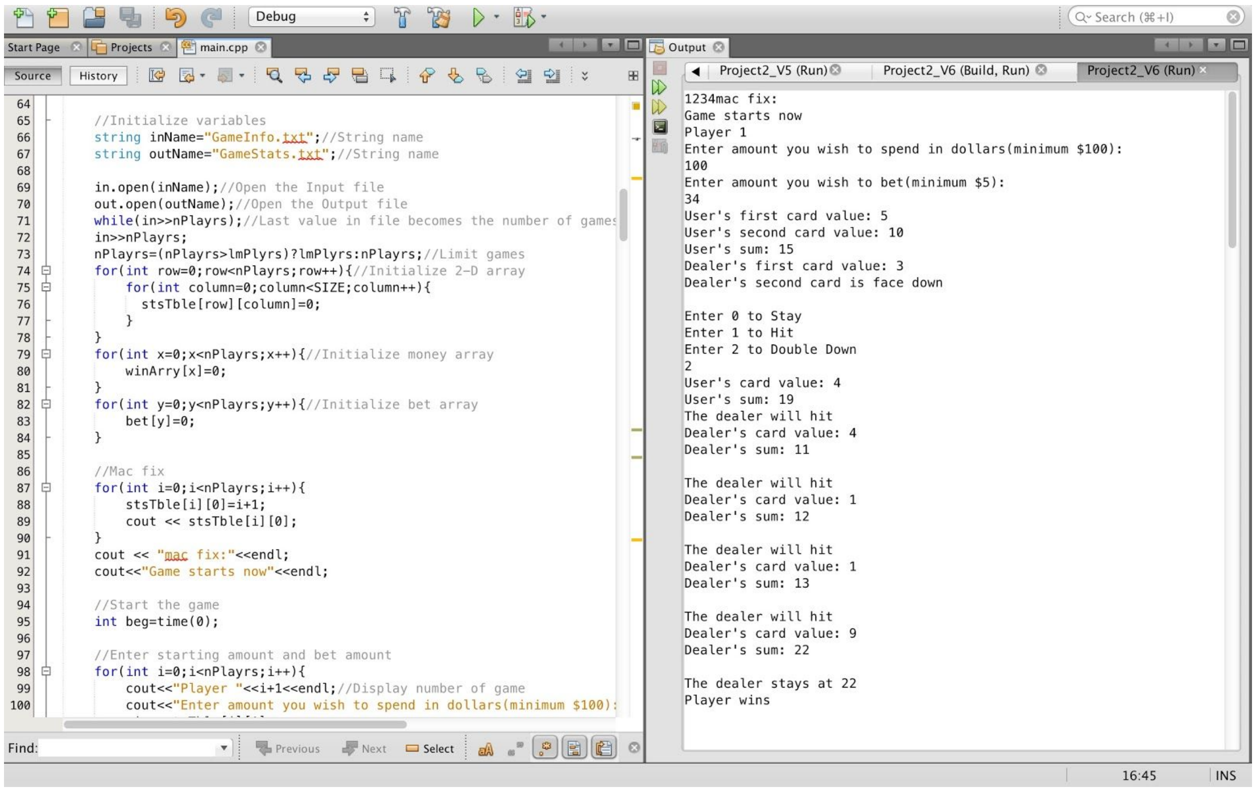
```
          //End case 1


          //End case 2


          //End case 3


         //End switch

         //Show dealer's card and sum
                //Dealer must stay at 17


                //Update dealer's sum


      //Output winner
                                                                     //Call function

                                //Separate the games

          //Re-initialize/Reset sums for next game


    //End the For loop

                   //End time of Game play

   //Output the game statistics to the screen




   //Output the game statistics to a file
```

```
    //Call bubble sort function for ranking

    //Output ranking to screen

    //Output ranking to file

    //Search amount a specific player has

    //Close files

    //Exit main

//crdVal function

//winner function
```

```
          //end nested if-else

                         //Update losses


                         //Update wins

                    //Update wins


                    //Update losses

          //Return from winner

//sort function
         //Declare variables

                                        //Sort rows. Swap both columns




          //Return from sort

//Search function
```

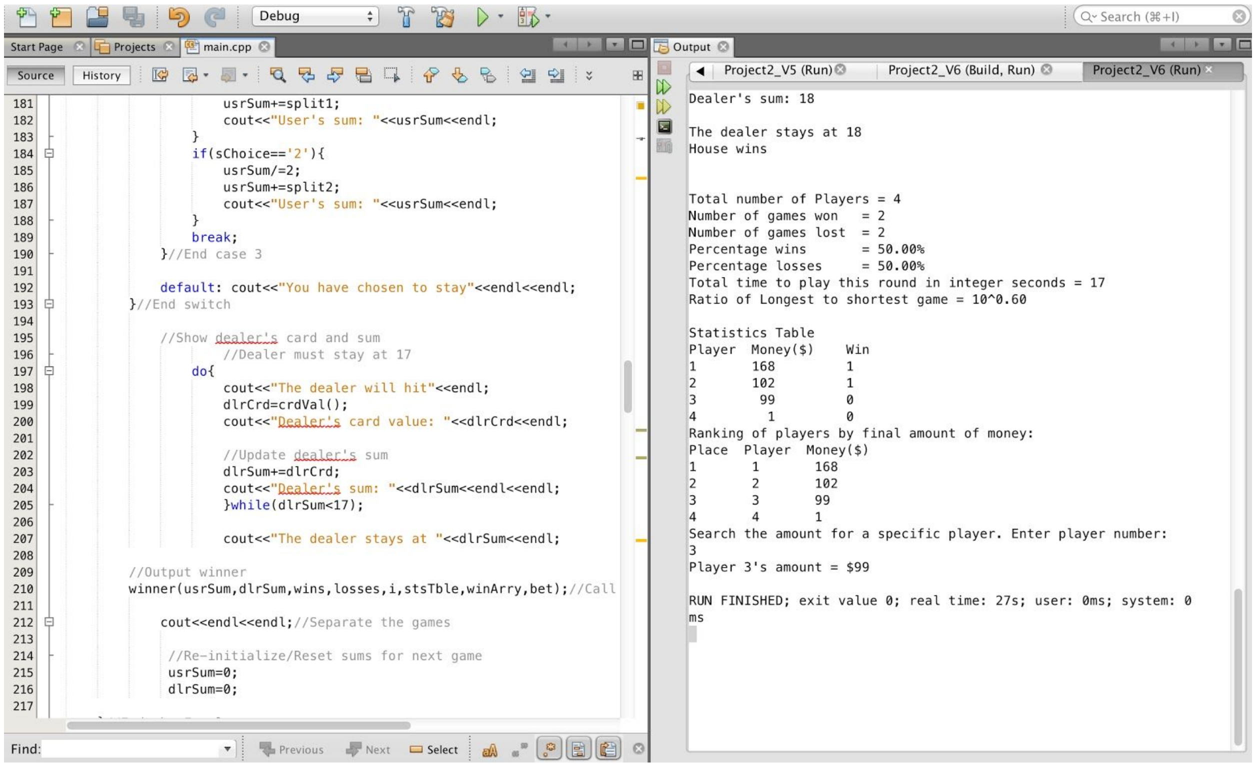
//Return from search

Proof of Working Program



```cpp

    //Initialize variables
    string inName="GameInfo.txt";//String name
    string outName="GameStats.txt";//String name

    in.open(inName);//Open the Input file
    out.open(outName);//Open the Output file
    while(in>>nPlayrs);//Last value in file becomes the number of games
    in>>nPlayrs;
    nPlayrs=(nPlayrs>lmPlyrs)?lmPlyrs:nPlayrs;//Limit games
    for(int row=0;row<nPlayrs;row++){//Initialize 2-D array
        for(int column=0;column<SIZE;column++){
          stsTble[row][column]=0;
        }
    }
    for(int x=0;x<nPlayrs;x++){//Initialize money array
        winArry[x]=0;
    }
    for(int y=0;y<nPlayrs;y++){//Initialize bet array
        bet[y]=0;
    }

    //Mac fix
    for(int i=0;i<nPlayrs;i++){
        stsTble[i][0]=i+1;
        cout << stsTble[i][0];
    }
    cout << "mac fix:"<<endl;
    cout<<"Game starts now"<<endl;

    //Start the game
    int beg=time(0);

    //Enter starting amount and bet amount
    for(int i=0;i<nPlayrs;i++){
        cout<<"Player "<<i+1<<endl;//Display number of game
        cout<<"Enter amount you wish to spend in dollars(minimum $100):
```

```
1234mac fix:
Game starts now
Player 1
Enter amount you wish to spend in dollars(minimum $100):
100
Enter amount you wish to bet(minimum $5):
34
User's first card value: 5
User's second card value: 10
User's sum: 15
Dealer's first card value: 3
Dealer's second card is face down

Enter 0 to Stay
Enter 1 to Hit
Enter 2 to Double Down
2
User's card value: 4
User's sum: 19
The dealer will hit
Dealer's card value: 4
Dealer's sum: 11

The dealer will hit
Dealer's card value: 1
Dealer's sum: 12

The dealer will hit
Dealer's card value: 1
Dealer's sum: 13

The dealer will hit
Dealer's card value: 9
Dealer's sum: 22

The dealer stays at 22
Player wins
```



```
                    usrSum+=split1;
                    cout<<"User's sum: "<<usrSum<<endl;
                }
                if(sChoice=='2'){
                    usrSum/=2;
                    usrSum+=split2;
                    cout<<"User's sum: "<<usrSum<<endl;
                }
                break;
            }//End case 3

            default: cout<<"You have chosen to stay"<<endl<<endl;
        }//End switch

            //Show dealer's card and sum
                    //Dealer must stay at 17
                do{
                    cout<<"The dealer will hit"<<endl;
                    dlrCrd=crdVal();
                    cout<<"Dealer's card value: "<<dlrCrd<<endl;

                    //Update dealer's sum
                    dlrSum+=dlrCrd;
                    cout<<"Dealer's sum: "<<dlrSum<<endl<<endl;
                    }while(dlrSum<17);

                    cout<<"The dealer stays at "<<dlrSum<<endl;

        //Output winner
        winner(usrSum,dlrSum,wins,losses,i,stsTble,winArry,bet);//Call

            cout<<endl<<endl;//Separate the games

             //Re-initialize/Reset sums for next game
             usrSum=0;
             dlrSum=0;
```

```
Dealer's sum: 18

The dealer stays at 18
House wins


Total number of Players = 4
Number of games won   = 2
Number of games lost  = 2
Percentage wins       = 50.00%
Percentage losses     = 50.00%
Total time to play this round in integer seconds = 17
Ratio of Longest to shortest game = 10^0.60

Statistics Table
Player  Money($)    Win
1       168         1
2       102         1
3        99         0
4         1         0
Ranking of players by final amount of money:
Place  Player  Money($)
1       1       168
2       2       102
3       3       99
4       4       1
Search the amount for a specific player. Enter player number:
3
Player 3's amount = $99

RUN FINISHED; exit value 0; real time: 27s; user: 0ms; system: 0
ms
```

Program

```
/*
 * File:   main.cpp
 * Author: Armando Gutierrez
 * Created on May 17, 2017, 9:28 PM
 */

//System Libraries
#include <iostream>  //Input - Output Library
#include <ctime>     //Time for rand
#include <cstdlib>   //Srand to set the seed
#include <iomanip>   //Format the output
#include <fstream>   //File I/O
#include <string>    //Strings
#include <cmath>     //Math functions

//User Libraries
```

```
using namespace std;

/*
 *
 */

//Global Constants
const float PERCENT=100.0f;//Conversion to Percent
const int HDTHSND=10e5; //One Hundred Thousand
const int SIZE=2;//Number of columns for 2-D array

//Function Prototypes
int crdVal ();//Returns new card value
void winner(int, int, int&, int&, int, int[][SIZE],int[], int[]);//Output winner
void sort(int[][SIZE],int);//Bubble sort
int search(int[][SIZE],int,int);//Binary search

//Executable code starts here
int main(int argc, char** argv) {

    //Set the random number seed
    srand(static_cast<unsigned int>(time(0)));

    //Declare file and game variables
    ifstream in;//Input File
    ofstream out;//Output File
    ofstream money;//Output File
    int dlrCrd, usrCrd;//Dealer's/User's card
    int dlrSum=0, usrSum=0;//Dealer's/User's sum of cards
    char choice;//User's choice
    int wins=0,losses=0;//Number of wins/losses
    int nPlayrs, lmPlyrs=HDTHSND;//Number of games/Limit to games
    int hand[10]={};//Cards in first hand
    int split1,split2;//The third cards in each hand for split option
    char sChoice;//Split choice
    int stsTble[nPlayrs][SIZE];//Statistics Table
    int winArry[nPlayrs];//Initial amount of money
    int bet[nPlayrs];//Bets
    int numPlyr;//Number of player
    int plyrAmt;//Player's amount

    //Initialize variables
    string inName="GameInfo.txt";//String name
    string outName="GameStats.txt";//String name

    in.open(inName);//Open the Input file
    out.open(outName);//Open the Output file
    while(in>>nPlayrs);//Last value in file becomes the number of games
    in>>nPlayrs;
    nPlayrs=(nPlayrs>lmPlyrs)?lmPlyrs:nPlayrs;//Limit games
    for(int row=0;row<nPlayrs;row++){//Initialize 2-D array
       for(int column=0;column<SIZE;column++){
         stsTble[row][column]=0;
       }
    }
    for(int x=0;x<nPlayrs;x++){//Initialize money array
       winArry[x]=0;
    }
    for(int y=0;y<nPlayrs;y++){//Initialize bet array
       bet[y]=0;
    }

    //Mac fix
    for(int i=0;i<nPlayrs;i++){
       stsTble[i][0]=i+1;
       cout << stsTble[i][0];
    }
    cout << "mac fix:"<<endl;
    cout<<"Game starts now"<<endl;

    //Start the game
    int beg=time(0);

    //Enter starting amount and bet amount
    for(int i=0;i<nPlayrs;i++){
       cout<<"Player "<<i+1<<endl;//Display number of game
       cout<<"Enter amount you wish to spend in dollars(minimum $100):"<<endl;
       cin>>stsTble[i][1];
       cout<<"Enter amount you wish to bet(minimum $5):"<<endl;
       cin>>bet[i];

       //User first card
       usrCrd=crdVal();//[1-11]
       cout<<"User's first card value: "<<usrCrd<<endl;
       hand[0]=usrCrd;
```

```
//User's second card
usrCrd=crdVal();//[1-11]
cout<<"User's second card value: "<<usrCrd<<endl;
hand[1]=usrCrd;

//Update and display user's sum
usrSum=hand[0]+hand[1];
cout<<"User's sum: "<<usrSum<<endl;

//Show dealer's card 1
dlrCrd=crdVal();//[1-11]
cout<<"Dealer's first card value: "<<dlrCrd<<endl;

//Update dealer's sum
dlrSum+=dlrCrd;

//Dealer's second card. Do not display
dlrCrd=crdVal();//[1-11]
cout<<"Dealer's second card is face down"<<endl<<endl;

//Update dealer's sum
dlrSum+=dlrCrd;

//Output menu
cout<<"Enter 0 to Stay"<<endl;
cout<<"Enter 1 to Hit"<<endl;
cout<<"Enter 2 to Double Down"<<endl;
if(hand[0]==hand[1])
cout<<"Enter 3 to Split"<<endl;

cin>>choice;//Enter user's choice to hit or stay

switch(choice){//Switch statement with 3 cases
   case '1': {
      bool hit=true;
      do{
      usrCrd=crdVal();
      cout<<"User's card value: "<<usrCrd<<endl;
      //Update and display user sum
      usrSum+=usrCrd;
      cout<<"User's sum: "<<usrSum<<endl;

      cout<<"Enter 1 to hit. Enter 0 to stay"<<endl;
```

```
        cin>>hit;//Enter user's choice to hit or stay

        }while(hit && usrSum<21 && dlrSum<21);

        break;
    }//End case 1

    case '2':{
        usrCrd=crdVal();
        cout<<"User's card value: "<<usrCrd<<endl;
        //Update and display user sum
        usrSum+=usrCrd;
        cout<<"User's sum: "<<usrSum<<endl;
        bet[i]*=2;//Double the bet

        break;
    }//End case 2

    case '3':{
        cout<<"You will be hit once per hand"<<endl;
        split1=crdVal();
        split2=crdVal();
        cout<<"Hand 1 card: "<<split1<<endl;
        cout<<"Hand 2 card: "<<split2<<endl;
        cout<<"Enter '1' or '2' to choose which hand you would like to keep"<<endl;
        cin>>sChoice;
        if(sChoice=='1'){
            usrSum/=2;
            usrSum+=split1;
            cout<<"User's sum: "<<usrSum<<endl;
        }
        if(sChoice=='2'){
            usrSum/=2;
            usrSum+=split2;
            cout<<"User's sum: "<<usrSum<<endl;
        }
        break;
    }//End case 3

    default: cout<<"You have chosen to stay"<<endl<<endl;
}//End switch

    //Show dealer's card and sum
```

```
            //Dealer must stay at 17
          do{
            cout<<"The dealer will hit"<<endl;
            dlrCrd=crdVal();
            cout<<"Dealer's card value: "<<dlrCrd<<endl;

            //Update dealer's sum
            dlrSum+=dlrCrd;
            cout<<"Dealer's sum: "<<dlrSum<<endl<<endl;
            }while(dlrSum<17);

            cout<<"The dealer stays at "<<dlrSum<<endl;

    //Output winner
    winner(usrSum,dlrSum,wins,losses,i,stsTble,winArry,bet);//Call function

        cout<<endl<<endl;//Separate the games

         //Re-initialize/Reset sums for next game
         usrSum=0;
         dlrSum=0;

}//End the For loop

int end=time(0);//End time of Game play

//Output the game statistics to the screen
cout<<fixed<<setprecision(2)<<showpoint;
cout<<"Total number of Players = "<<nPlayrs<<endl;
cout<<"Number of games won   = "<<wins<<endl;
cout<<"Number of games lost  = "<<losses<<endl;
cout<<"Percentage wins       = "
        <<static_cast<float>(wins)/nPlayrs*PERCENT<<"%"<<endl;
cout<<"Percentage losses     = "
        <<static_cast<float>(losses)/nPlayrs*PERCENT<<"%"<<endl;
cout<<"Total time to play this round in integer seconds = "<<end-beg<<endl;
cout<<"Ratio of Longest to shortest game = 10^"<<log10(nPlayrs)<<endl<<endl;
cout<<"Statistics Table"<<endl;
cout<<"Player  Money($)    Win"<<endl;
for(int row=0;row<nPlayrs;row++){
    for(int column=0;column<SIZE;column++){
        cout<<stsTble[row][column]<<setw(10);
    }
    cout<<winArry[row]<<endl;
}

//Output the game statistics to a file
out<<fixed<<setprecision(2)<<showpoint;
out<<"Total number of Players = "<<nPlayrs<<endl;
out<<"Number of games won   = "<<wins<<endl;
out<<"Number of games lost  = "<<losses<<endl;
out<<"Percentage wins       = "
        <<static_cast<float>(wins)/nPlayrs*PERCENT<<"%"<<endl;
out<<"Percentage losses     = "
        <<static_cast<float>(losses)/nPlayrs*PERCENT<<"%"<<endl;
out<<"Total time to play this round in integer seconds = "<<end-beg<<endl;
out<<"Ratio of Longest to shortest game = 10^"<<log10(nPlayrs)<<endl<<endl;
out<<"Statistics Table"<<endl;
out<<"Player  Money($)    Win"<<endl;
for(int row=0;row<nPlayrs;row++){
    for(int column=0;column<SIZE;column++){
        out<<stsTble[row][column]<<setw(10);
    }
    out<<winArry[row];
    out<<endl;
}

//Call bubble sort function for ranking
sort(stsTble,nPlayrs);

//Output ranking to screen
cout<<"Ranking of players by final amount of money:"<<endl;
cout<<"Place  Player  Money($)"<<endl;

for(int row=0;row<nPlayrs;row++){
    cout<< row+1<<"       ";
    for(int column=0;column<SIZE;column++){
        cout << stsTble[row][column]<<"       ";
    }
    cout<<endl;
}

//Output ranking to file
out<<"Ranking of players by final amount of money:"<<endl;
out<<"Place  Player  Money($)"<<endl;
```

```
    for(int row=0;row<nPlayrs;row++){
        out<< row+1<<"       ";
        for(int column=0;column<SIZE;column++){
            out << stsTble[row][column]<<"       ";
        }
        out<<endl;
    }


    //Search amount a specific player has
    cout<<"Search the amount for a specific player. Enter player number:"<<endl;
    cin>>numPlyr;
    plyrAmt=search(stsTble,nPlayrs,numPlyr);
    cout<<"Player "<<numPlyr<<"'s amount = $"<<plyrAmt<<endl;

    //Close files
    in.close();
    out.close();

    //Exit main
    return 0;
}


//crdVal function
int crdVal (){
    int card;
    card=1+(rand()%10);
    return card;
}

//winner function
void winner(int usrSum, int dlrSum, int& wins, int&losses,
                int i, int stsTble[][SIZE],int winArry[], int bet[]){

    if(usrSum<21 && dlrSum<21){
            if(usrSum==dlrSum)
             cout<<"Push"<<endl;
            else if (usrSum>dlrSum){
                cout<<"Player wins"<<endl;
                wins++;
                stsTble[i][1]+=bet[i];
                winArry[i]=1;
            }
            else{
                cout<<"House wins"<<endl;
                losses++;
                stsTble[i][1]-=bet[i];
                winArry[i]=0;
            }
            }//end nested if-else

        if(usrSum>21){
            cout<<"Player busts!"<<endl;
            losses++;//Update losses
            stsTble[i][1]-=bet[i];
            winArry[i]=0;
            }

        if(dlrSum>21 && usrSum<21){
            cout<<"Player wins"<<endl;
            wins++;//Update wins
            stsTble[i][1]+=bet[i];
            winArry[i]=1;
            }

        if(usrSum==21){
            cout<<"Player gets blackjack!"<<endl;
            wins++;//Update wins
            stsTble[i][1]+=bet[i];
            winArry[i]=1;
        }

        if(dlrSum==21 && usrSum<21){
            cout<<"House gets blackjack!"<<endl;
            losses++;//Update losses
            stsTble[i][1]-=bet[i];
            winArry[i]=0;
        }
    //Return from winner
}

//sort function
void sort(int stsTble[][SIZE],int nPlayrs){
    //Declare variables
    bool swap;
    int temp,t2;

        do{
            swap=false;
            for(int i=0;i<nPlayrs-1;i++){
                if(stsTble[i][1]<stsTble[i+1][1]){
                    temp=stsTble[i][1];//Sort rows. Swap both columns
                    t2=stsTble[i][0];
                    stsTble[i][1]=stsTble[i+1][1];
                    stsTble[i][0]=stsTble[i+1][0];
                    stsTble[i+1][1]=temp;
                    stsTble[i+1][0] = t2;
                    swap=true;
                }
            }
        }while(swap);
    //Return from sort
}
//search functions
int search(int stsTble[][SIZE],int nPlayrs,int numPlyr){
    int first=0;
    int last=nPlayrs-1;
    int middle;
    int place= -1;
    bool found=false;

    while (!found && stsTble[first][0]<=stsTble[last][0]){
        middle=(first+last)/2;
        if(stsTble[middle][0]==numPlyr)
        {
            found=true;
            place=middle;
        }
        else if(stsTble[middle][0]>numPlyr){
            last=middle-1;
        }
        else{
            first=middle+1;
        }
    }
    return stsTble[place][1];//Return from search
}
```