Project 2

<Heads Up Limit Texas Hold'em>

CSC-5 43952
Branden Hitt
Date: 6/7/2015

Table of Contents

Introduction

Title: Heads Up Limit Texas Hold'em (Poker Variation)

This is a small program for heads up (1 on 1) limit ($5 max bet) Texas Hold'em.

At the beginning of the program, the user will be prompted to choose if they want the rules displayed or not. The rules also include the hand rankings which you can find below. From there they are asked to give their name. After that, they are introduced to the AI competition, and they begin a round of poker with a buying of $50.

To begin a round, the players must first pay the Ante (forced $5 bet per hand). Following that, the program deals out a hand of 2 cards to each player (hole cards). There is a round of betting. Next, 3 community cards will be dealt to the center, called the Flop. (Community cards will be used at the end in combination with/without your beginning hole cards to find your best hand). Next, is another round of betting. Then, a 4th community card called the Turn will be dealt to the center. Again, another round of betting. Finally, the 5th and last community card called the River will be dealt to the center. The user is told his best hand of any 5 cards (out of 7) and its value (pair, high card, flush, etc) along with being prompted for a final round of betting. Now that both players have their best hand and betting is finished, the program will find the best hand between the players and reward the Pot to the winner. If both players still have money to bet, another round will begin.

Once a player takes all of the money from the other player, the game will finish and he will be declared the winner. At the very end of the program, the results are submitted to a file called Result.dat. This records the player name, hands won, total rounds, percent win, and the cash they won or lost.

Ex: “Congrats to Branden H for being the table's big winner!

Out of 6 total hands, Branden won 3 hands of Poker.

Thank you for playing!”

The final results for the player are then sent to a file (Results.dat)

Ex: “Cash Win: $50

Player Name: Branden

Round Wins : 2

Total Rounds: 6

Percent Win: %33.33”

Betting System

The betting for my program was difficult enough to be a challenge yet not too complicated for the player. I knew right off the bat that I would want an Ante (forced $5 bet per hand) so that the player could not just fold until he had a great hand. So after I got that part figured out, I still needed to decide how bets would work. I ended up having 3 options per bet.

1.check/call..... If there is no bet, you may pass by checking. If there is a bet, you may call it and the round of betting is over.

2.bet/raise..... If there is no bet, you may bet $5 and the other player must match it. If there is a bet, you can choose to add $5 to it and the player must match the difference.

3.fold..... This option allows you to forfeit your hand so that you do not lose further money. However, you will lose that pot regardless of who has the better hand.

The AI has the same choices as the player for betting, however he mostly picks at random. He is most likely to call, but will not fold if there is no current bet. It should be noted that if a player folds or is “All-In” (no more money left to bet), the program will skip to the end and not take further bets.

The most trouble I had with implementing the betting system would be to make sure that all of the logic worked correctly. More specifically, when it got down to an All-In, I had to make sure no one would go negative and that the other player would match correctly. I also had problems with getting the program to continue betting if the other player raised. Finally, from what I remember the biggest problem was to make sure that I

was taking away the correct amount from them including what they had already put in the Pot. Overall, I believe I fixed all of the logic unless a player decides to input something other than the choices.

Hand Rankings

(Low) **High Card**: This hand has nothing to rely on but a High card.

Ex: 3 Diamonds | Jack Clubs | 8 Spades | 4 Hearts | 2 Spades

Pair: Two cards of the same rank.

Ex: Ace Hearts | Ace Diamonds | 8 Clubs | 4 Spades | 7 Hearts

Three of a Kind: Three cards of the same rank.

Ex: 7 Clubs | 7 Diamonds | 7 Spades | King Clubs | 3 Diamonds

Straight: Five cards in a sequence, but not of the same suit.

Ex: 5 Hearts | 6 Diamonds | 7 Spades | 8 Diamonds | 9 Clubs

Flush: Any five cards of the same suit, but not in a sequence.

Ex: 4 Spades | Jack Spades | 8 Spades | 2 Spades | 9 Spades

Full House: Three of a Kind and a Pair.

Ex: 10 Hearts | 10 Diamonds | 10 Spades | 9 Clubs | 9 Diamonds

Four of a Kind: All four cards of the same rank.

Ex: Jack Hearts | Jack Spades | Jack Diamonds | Jack Clubs | 7 Hearts

Straight Flush: Five cards in a sequence, all in the same suit.

Ex: 4 Spades | 5 Spades | 6 Spades | 7 Spades | 8 Spades

(High) **Royal Flush**: 10, J, Q, K, A (all in the same suit).

Ex: 10 Spades | Jack Spades | Queen Spades | King Spades | Ace Spades

In the event of a tie hand (ex. both players have a pair), the higher pair is chosen. In the event that both players have the same pair, then it will find who has the High card win. This applies to almost all cases that could arise however I stopped after one high card compare with each hand. This was an improvement I made to the last program since I didn't figure out yet how to find the better hand after a pair.

Summary

Project size: 1101 lines (with comments)

Number of Variables: ~94

Number of Functions: 22 (+1 for main)

For the second project I decided to build upon my first, but also make it its own game. This time I made it the Texas Hold'em variation of poker which had a lot more logic to figure out as well as implementing a betting system. After all, what is poker without some betting?

I tried to include everything we have covered so far, and I believe I fit it all in the program except for maybe abs() function since it wasn't really needed. Functions were extremely useful for saving time and also for organization. The most useful thing might have been if()else() statements, since there ended up being a lot of logic for the program. Arrays were very useful to hold the deck that I used in the game. Finally, I utilized sorting to sort the hands when looking for a high card or straight.

I am very satisfied with how my second program came out. I am more than sure that I could have used arrays better if I had started from scratch, however, I felt it easier to just build on what I currently had. The organization/efficiency could also be improved, yet it all seems to run great after debugging it for days and accounting for rare exceptions.

Future Improvements

There are many things I would love to add to this game to improve it. Of course, having sound and graphics for the game would be nice, but that is a bit out of my reach as I am new to programming. I previously mentioned how it could be more efficient or utilize arrays better so that could be a fix. Still, the greatest improvement I would want for this program is for it to have a better AI for the computer rather than the mostly random AI that already exists. This could be extremely difficult to implement since every round of betting would have a different amount of cards to base its bet on and I have no idea how to calculate the odds. In addition, having multiple AI to go against would be great but might compound the difficulty and work going into it. Lastly, I

would want to make sure every bit of logic in the program was resilient.

Overall, I am extremely happy with the improvements that I met from the last program. Looking back, I accomplished the further accuracy aim on hands, as well as changing the variation to texas hold'em, and implementing a betting system.

Simple Program Function Overview

main(

iniDeck(); ← create 52 card deck

rules(); ← display rules/hand rankings if user wants

round(); ← start a new round

iniDeck(); ← shuffle deck for new round

)

round(

worth(); ← display players cash amounts

ante(); ← collect the ante from both players

worth(); ← display players cash amounts

deal1(); ← this is done for both players hole cards (2 per player)

bet(); ← round of betting

deal1(); ← deal 3 community cards (Flop)

bet(); ← round of betting

deal 1(); ← deal 1 community card (Turn)

bet(); ← round of betting

deal1(); ← deal last community card (River)

bestCom(); ← find both players best hand combination from the 7 cards

finCom(); ← set the final hands

hiCard(); ← sorts the cards for hi card and ordered display

ranking(); ← find hand rankings number for all players

sRank(); ← gives string for output of hand rank (ex: Pair)

carVal(); ← gives string for individual card/ used for display to player (ex: 9 hearts)

bet(); ← last round of betting'

worth(); ← display players cash amounts

winHand(); ← find best hand

winner(); ← find string for winner name

award(); ← give pot to winner

)

bet(

playBet(); ← players turn to bet (if they are first)

check(); ← check to see if betting is concluded

comBet(); ← computer turn to bet

check(); ← check to see if betting is concluded

playBet(); ← players turn to bet (if they are second)

check(); ← check to see if betting is concluded

)

Psuedocode

```
/*
 * File:   main.cpp
 * Author: Branden Hitt
 * Created on May 6, 2015, 1:02 PM
 * Notes for next update: work on betting format, 3rd high card, all-Ins
 *      Purpose: Head's Up Texas Hold'em
 */
//System Libraries
//I/O standard
//computer time
//for random seed
//formatting
//for strings
//file in/out

//User Libraries
```

```
//Global Constants
//Function Prototypes
//instructions on how to play
//initializes and shuffles the deck
//starts a new round of poker
//deal 1 card
//gives hand value
 //finds a short string rank of hand
//finds what the card is for output
//sort for high card
//find best version of your hand
//set hand with best combination
//find winning hand between two players
//find round winner
//displays money totals
// calls ante
//individual ante
// main betting function
//player bet function
//computer bet function
//check to end betting round
//award pot to winner
//split the pot for ties
//all in or fold for skip
//Execution Begins Here!

   //Declare Variables
   //output to file
   //open file
   //set random seed
   //input for info query, and repeat query
   //total wins for each player
   //beginning money total
   //dealer button
   //overall winner and player name input
   //size of the deck
   //declare array with zeros
   //current round win
   //percentage of wins for the player/ totals rounds
   //fill array with card values (1-52)
   //Prompt for rules
   //prompt for name of player
   //do
      //Start new round
      //if 1 player 1 win round
      //if 2 player 2 win round
      //Shuffle
      //increment the total rounds
   //while player wants to repeat
   //Congratulate Overall Winner
   //output results to a file
   //Exit Stage Right!
//*****************************************//
//*           Rules Function              *//
//*****************************************//
   //declare variables
   //menu / choice
   //bool for skip
   //begin explanation of how to play
```

```
    //do
        //start showing hand rankings
        //prompt for specific hand ranking display
        //menu for hand display
            //case for hi card
            //case for pair
            //case for two pair
            //case for three of a kind
            //case for straight
            //case for flush
            //case for full house
            //case for four of a kind
            //case for straight flush
            //case for royal flush
            //default for skip
            //set skip to true
        //if skip is false
        //prompt for repeat
    //while player wants to repeat
//*********************************************//
//*           Initialize Deck               *//
//*********************************************//
    //increment while still in array
        //for position i, set i to i ex: 41 =41
//*********************************************//
//*              New round                  *//
//*********************************************//
    //Declare variables
    //player hole cards
    //computer hole cards
    //community cards
    //number for combination of hand
    //player final hand cards 1-5
    //computer final hand cards 1-5
    //hand values
    //high card for tie hands
    //player number who has top hand
    //quick string for hand ranking
    //variables for betting
    //pot total
    //skip if there is all in
    //player or computer fold
    //get ante
    //display totals
    //Deal hole cards
    //display hole cards
    //call betting round 1
    //Deal 3 Community Cards (Flop)
        //betting round 2
    //Deal 1 Community Card (Turn)
        //betting round 3
    //Deal 1 Community Card (River)
    //find best combination of cards per player
    //set final hand to best combination
    //high card and sort
    //find hand value of player
    //find hand value of computer
    //set string for short hand ranking
    //display final community cards
```

```
    //state player hand value
    //Display player hand
        //final bet round
    //Unveil all hands
    //find winning hand
        //find winNum
    //find winner name
    //state winner
    //award pot to winner
    //display player totals
    //increment button
    //return winner number
//*********************************************//
//*             Deal 1                    *//
//*********************************************//
    //declare variables
    //repeat
    //new card
    //do
        //pick random number
        //if card is 0, then its already picked, so repeat
        //else dont repeat
    //while need new card
    //set the card you picked to zero in array
    //c1 = card
//*********************************************//
//*           Card Values                 *//
//*********************************************//
    //declare string f for card name
    //Spades
    //Hearts
    //Clubs
    //Diamonds
    //return card name
//*********************************************//
//*           Hand Ranking                *//
//*********************************************//
    //declare variables
    //high card hand value= 1
    //temp
    // straight = n, flush = n
    //flush = 6
    //spades
    //hearts
    //clubs
    //diamonds
    //get rid of suits
    //sort for straight
        //sort first value
        //sort second value
        //sort third value
        //sort fourth value
    //high card for flush
    //high card for high card hand
    //pair = 2
    //two pair = 3
    //three of a kind = 4
    //straight = 5
    //full house = 7
```

```
   //four of a kind = 8
   //straight flush = 9
   // if flush and straight then value =9
   //return hand value
//*********************************************//
//*            Short Hand Rank               *//
//*********************************************//
   //declare string for hand type
   //if 1 then high card
   //if 2 then pair
   //if 3 then two pair
   // if 4 then 3 of a kind
   //if 5 then straight
   //if 6 then flush
   // 7 then full house
   //if 8 then four of a kind
   // if 9 then straight flush
   //return string for hand type
//*********************************************//
//*              High Card/Sort              *//
//*********************************************//
   //temp variables for swap
   //get rid of suits
   //take away 13 for values above 13
   //sort first value
      //swap if needed
   //sort second value
      //swap if needed
   //sort third value
      //swap if needed
   //sort fourth value
      //swap if needed
//*********************************************//
//*           Best Combination              *//
//*********************************************//
   //variables
   //high,temporary,placeholder,best combination number,current High Card
   //5 community cards
   //3 com cards and 2 hole
   //4 com card and first hole card
   //4 com card and second hole card
   //return best combination found
//*********************************************//
//*              Set Final Hand              *//
//*********************************************//
   // if the comb =  number
   // set specific hand
//*********************************************//
//*              Winning Hand                *//
//*********************************************//
   //find top hand value
   //declare variables for ties
   //find all that have the same value of the top hand
   //compare ties for top hand
   //if two have tie
      //win =1
      //win =2
   //if no tie
   //win 1
```

```
   //win 2
   //return win
//*********************************************//
//*          Winning Player                   *//
//*********************************************//
   //declare string variable for winner
   //if 1 then winner = user
   //if 2 then winner = tom
   // else tie game
   //return winner
//*********************************************//
//*              Worth                        *//
//*********************************************//
   //state string
//*********************************************//
//*              AnteUp                       *//
//*********************************************//
   //variables
   //call indAnte
   //player 1 ante
   //player 2 ante
   //find if skip is needed
//*********************************************//
//*          Individual Ante                  *//
//*********************************************//
   //if a=0 fold is true
   //if a>0 take away ante and add to pot
   //if a=0 && fold=false then its an all in
//*********************************************//
//*          Betting Function                 *//
//*********************************************//
   //declare variables
   //tally to make sure both players bet at least once
   //player/computer total bet and current bet
   //do
        //call player bet
     //call computer bet
        //call player bet
     //check for raises
   //while stoop = false
   //find skip
   //display totals
//*********************************************//
//*          Player Bet                       *//
//*********************************************//
   //variables
   //display current bet
   //prompt for bet
//*********************************************//
//*           Computer Bet                    *//
//*********************************************//
   //variables
   //find computer bet
   //do computer bet
//*********************************************//
//*         Check for Bet end                 *//
//*********************************************//
   //if both players have bet the same return true
   //if one player has folded return true
```

```
   //if a player is all in return true
   //otherwise return false
//*****************************************//
//*        Award pot to winner            *//
//*****************************************//
   //winner gains the pot
//*****************************************//
//*          Split the pot                *//
//*****************************************//
   //divide pot by two
   //award both players half the pot
//*****************************************//
//*          All in or Fold               *//
//*****************************************//
   //if either player is all in or fold return true
   //else return false
```

Concepts from the Book (Savitch 9th Ed)

Chapter	Concept	Description	Location
1. Introduction	#include <iostream>	System library for input and output	Line 10
	using namespace std;	Used standard naming	Line 16
	Int main( ){ }	Main function	Line 47
	char info,repeat;	Variable declaration	Line 57
	cout<<	Output	Line 70
	cin>>	Input	Line 74
	char	Character type variable	Line 57
2. Basics	int	Int type variable	Line 68
	int totR	identifier	Line 68
	x=y*z	Assignment statement	Line 114
	endl;	End line	Line 113
	float	Floating point type variable	Line 68
	short	Short Integer	Line 59
	#include <string>	String library	Line 14
	string player1;	String variable	Line 61
	bool	True or false variable	Line 60
	* / operators	Multiply or divide	Line 114
	+- operators	Add or Subtract	Line 999
	% operator	Mod operator for remainder	Line 122
	if()else()	If-else bool statements	Line 117

	==	Comparison Operator “equal to”	Line 97
	>,<,>= ,<=	Comparison Operators less than,greater than, less than equal to, greater than equal to	Line 102
	&&	And Operator	Line 102
	||	Or Operator	Line 75
	do{}while();	Do something while condition is satisfied (do-while loop)	Line 94
	i++,i--;	Increment, decrement operators	Line 260
	//something	Comment	Line 242
	const short SIZE=53;	Constant variable	Line 62
	#include <iomanip>	Library for formatting	Line 13
	fixed<<setprecision(2)<< showpoint	Formatting for numbers	Line 116
3. Multi-way Branches	If(if ())	Multi-way if statements	Line 719

	switch(menu){
	cases
	}
		Switch statement for menu	Line 191

	for(i;i<n;i++)	For loop	Line 531
	break;	Leaves a loop or switch	Line 195
4. Procedural Abstraction and Functions that return a value	#include <ctime>	ctime library for computer time	Line 11
	srand();	Set the random seed	Line 56
	rand();	Get random number	Line 444
	static_cast<unsigned int>	Type cast	Line 56
	#include <cstdlib>	Library used for random seed	Line 12
	short round();	Function that returns a value	Line 25
	void hiCard(short &);	Function with a pass-by-reference	Line 30
	void loose(char [] [2],bool=true);	Defaulted Parameter	Line 45
5. Functions for all Subtasks	void rules();	Void function	Line 23

	return;	Ends function	Line 126
	string aiRank(short);	Call by value in function	Line 28
6. Streams and Basic I/O	ofstream fout;	Declares out file stream variable	Line 50
	#include <fstream>	fstream library for file usage	Line 15
	fout.open	Open file	Line 51
	fout.close	Close file	Line 123
	ifstream fin;	Declares in file stream variable	Line 49
	fin.open	Open file	Line 51
7. Intro to Arrays	short deck[Size]={};	Array declaration	Line 57
	void deal1(short, short [])	Array in a function	Line 26
	char ends[2][2];	Multi-Dim array	Line 64

References

1.Textbook - Problem Solving with C++, Savitch 9th Ed
2. PokerStars.com – (For card hand rankings list)

Flowcharts

Flowcharts of some of the examples can be found in the Flowcharts folder

Program Code

```
/*
    * File: main.cpp
    * Author: Branden Hitt
    * Created on May 6, 2015, 1:02 PM
    * Notes for next update:
    * Purpose: Head's Up Texas Hold'em
    */

    //System Libraries
    #include <iostream>//I/O standard
    #include <ctime>//computer time
    #include <cstdlib>//for random seed
    #include <iomanip>//formatting
    #include <string>//for strings
    #include <fstream>//file in/out
    using namespace std;

    //User Libraries

    //Global Constants

    //Function Prototypes
    void rules();//instructions on how to play
    void iniDeck(short [],short);//initializes and shuffles the deck
    short round(string ,short [],short &,short &,bool &);//starts a new round of poker
    void deal1(short &,short []);//deal 1 card
```

```
short ranking(short ,short ,short ,short ,short,short &,short &);//gives hand value
string sRank(short); //finds a short string rank of hand
string cardVal(short);//finds what the card is for output
void hiCard(short &,short &,short &,short &,short &);//sort for high card
short bestCom(short,short,short,short,short,short,short);//find best version of your hand
void finCom(short &,short &,short &,short &,short
&,short,short,short,short,short,short,short,short);//set hand with best combination
short winHand(short,short,short,short,short,short);//find winning hand between two players
string winner(string,short);//find round winner
void worth(short,short,short);//displays money totals
bool anteUp(short &, short &, short &);// calls ante
bool indAnte(short &,short &);//individual ante
bool bet(short &,short &, short &,bool,bool &,bool &);// main betting function
void playBet(short &,short &,short &,short &,bool &,short);//player bet function
void compBet(short &,short &,short &,short &,bool &,short);//computer bet function
bool check(short,short,short,bool &,bool &,short,short);//check to end betting round
void award(short &,short);//award pot to winner
void split(short &,short &,short &);//split the pot for ties
bool cease(short,short,bool,bool);//all in or fold for skip
void loose(char [][2],bool=true);//fill 2d array
//Execution Begins Here!
int main(int argc, char** argv) {
//Declare Variables
ifstream fin;
ofstream fout;//output to file
fout.open("Results.Dat");//open file
fin.open("Greet.text");//open greeting file
string greet;
getline(fin,greet);
cout<<greet<<endl;
srand(static_cast<unsigned int>(time(0)));//set random seed
char info,repeat;//input for info query, and repeat query
short p1Rw=0,p2Rw=0;//total wins for each player
short pl1M=50,pl2M=50;//beginning money total
bool button=true;//dealer button
string winner, player1;//overall winner and player name input
const short SIZE=53;//size of the deck
bool lobo=true;//bool for 2d array
char ends[2][2];//declare 2d array
loose(ends,lobo);//fill array
short deck[SIZE]={};//declare array with zeros
short rWin=0;//current round win
float perWin; int totR=0;//percentage of wins for the player/ totals rounds
iniDeck(deck,SIZE);//fill array with card values (1-52)
cout<<"This is a table for Heads Up Limit Texas Hold'em."<<endl;
//Prompt for rules
cout<<"If you would like instructions on how to play, enter in 'I' now:"<<endl;
cout<<"otherwise, enter in 'N'"<<endl;
cin>>info;
if(info=='I'||info=='i') rules();
//prompt for name of player
```

```
cout<<"************************"<<endl;
cout<<"Please first enter in your name:"<<endl;
cin.ignore();
getline(cin,player1);
cout<<"************************"<<endl;
cout<<"Hello, "<<player1<<"! Welcome to the table."<<endl;
cout<<"Please hit enter to begin"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
cout<<"Now that you are seated, why don't you meet your opponent."<<endl;
cout<<"Across from you is poker legend, Phil Hellmuth."<<endl;
cout<<"Phil is a 13 time World Series of Poker Champion with"<<endl;
cout<<"$18,282,014 in winnings!"<<endl;
cout<<"Please hit enter to continue"<<endl;
cin.get();
cin.ignore();
do{//do
//Start new round
rWin=round(player1,deck,pl1M,pl2M,button);
if(rWin==1) p1Rw++;//if 1 player 1 win round
if(rWin==2) p2Rw++;//if 2 player 2 win round
//Shuffle
iniDeck(deck,SIZE);
totR++;//increment the total rounds
}while(pl1M>0&&pl2M>0);//while player wants to repeat
//Congratulate Overall Winner
cout<<"************************"<<endl;
short highRW=p1Rw;
if(pl2M==0)winner=player1;
if(pl1M==0)winner="Phil Helmuth ", highRW=p2Rw;
cout<<"Congrats to "<<winner<<" for being the table's big winner."<<endl;
cout<<winner<<" takes home $100"<<endl;
cout<<"Out of "<<totR<<" total hands, "<<winner<<" won "<<highRW<<" hands of Poker."<<endl;
cout<<"Thank you for playing!"<<endl;
cout<<endl;
cout<<"(Your results can be found in Results.Dat)"<<endl;
perWin=((p1Rw*1.0f)/totR)*100.0f;
//output results to a file
fout<<fixed<<setprecision(2)<<showpoint;
if(winner==player1)fout<<"Cash Win: $50"<<endl;
else fout<<"Cash Loss: $50"<<endl;
fout<<"Player name : "<<player1<<endl;
fout<<"Hand Wins : "<<p1Rw<<endl;
fout<<"Total Rounds: "<<totR<<endl;
fout<<"Percent Win : %"<<perWin;
fout.close();
fin.close();
//Exit Stage Right!
return 0;
}
```

```
//*****************************************//
//* Rules Function *//
//*****************************************//
void rules(){
//declare variables
char menu,cho='N';//menu / choice
bool skip=false;//bool for skip
cout<<"************************"<<endl;
cout<<"How to Play:"<<endl;//begin explanation of how to play
cout<<"Step1:"<<endl;
cout<<"First both players pay an ante (forced $5 bet to join the round)"<<endl;
cout<<"Each player is dealt 2 cards (hole cards) from a standard 52-card deck."<<endl;
cout<<"Then each player will have a chance to bet/fold."<<endl;
cout<<"Note that a player may fold at any time during the game but will lose the pot."<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
cout<<"Step 2:"<<endl;
cout<<"From there, 3 community cards will be dealt to the center called the flop."<<endl;
cout<<"(Community cards will be used in combination with/without any of your hole cards"<<endl;
cout<<"to determine your final hand)"<<endl;
cout<<"Again, there will be a round of betting"<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
cout<<"Step 3:"<<endl;
cout<<"Next, 1 more community card will be dealt called the turn."<<endl;
cout<<"Another round of betting."<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
cout<<"Step 4:"<<endl;
cout<<"Finally, 1 final community card will be dealt to the center called the river."<<endl;
cout<<"Now that all 5 community cards are on the table, you can use any combination of them"<<endl;
cout<<"with/without your hole cards to get the best hand you can."<<endl;
cout<<"There is one final round of betting."<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
cout<<"Whoever has the best combination of cards based on rankings will win the round/pot."<<endl;
cout<<"The more rare the hand, the higher the ranking will be."<<endl;
cout<<"The worst hand is nothing special with just a High Card, while the best is a Royal Flush."<<endl;
cout<<"In the event that two players have the same hand, the higher card (pair,flush,etc)"<<endl;
```

```
cout<<"will win."<<endl;
do{//do
cout<<"The Hand Rankings are as followed:"<<endl;//start showing hand rankings
cout<<"0.High Card | 1.Pair | 2.Two Pair | 3.Three of a Kind | 4.Straight "<<endl;
cout<<"5.Flush | 6.Full House | 7.Four of a Kind | 8.Straight Flush | 9.Royal Flush"<<endl;
cout<<"************************"<<endl;
cout<<"To view an example/explanation of a specific hand, enter in"<<endl;
cout<<"the corresponding number. (for example: enter in 4 to view a straight)"<<endl;
cout<<"Otherwise, enter in S to get started:"<<endl;
cout<<"************************"<<endl;
cin>>menu;//prompt for specific hand ranking display
cout<<"************************"<<endl;
switch(menu){//menu for hand display
case'0':{//case for hi card
cout<<"High Card: This hand has nothing to rely on but a High card."<<endl;
cout<<"Ex: 3 Diamonds | Jack Clubs | 8 Spades | 4 Hearts | 2 Spades"<<endl;
break;
}
case'1':{//case for pair
cout<<"Pair: Two cards of the same rank."<<endl;
cout<<"Ex: Ace Hearts | Ace Diamonds | 8 Clubs | 4 Spades | 7 Hearts"<<endl;
break;
}
case'2':{//case for two pair
cout<<"Two Pair: Two different pairs of cards."<<endl;
cout<<"Ex: 4 Spades | 4 Clubs | 3 Clubs | 3 Diamonds | Queen Clubs"<<endl;
break;
}
case'3':{//case for three of a kind
cout<<"Three of a Kind: Three cards of the same rank."<<endl;
cout<<"Ex: 7 Clubs | 7 Diamonds | 7 Spades | King Clubs | 3 Diamonds"<<endl;
break;
}
case'4':{//case for straight
cout<<"Straight: Five cards in a sequence, but not of the same suit."<<endl;
cout<<"Ex: 5 Hearts | 6 Diamonds | 7 Spades | 8 Diamonds | 9 Clubs"<<endl;
break;
}
case'5':{//case for flush
cout<<"Flush: Any five cards of the same suit, but not in a sequence."<<endl;
cout<<"Ex: 4 Spades | Jack Spades | 8 Spades | 2 Spades | 9 Spades"<<endl;
break;
}
case'6':{//case for full house
cout<<"Full House: Three of a Kind and a Pair."<<endl;
cout<<"Ex: 10 Hearts | 10 Diamonds | 10 Spades | 9 Clubs | 9 Diamonds"<<endl;
break;
}
case'7':{//case for four of a kind
cout<<"Four of a Kind: All four cards of the same rank."<<endl;
cout<<"Ex: Jack Hearts | Jack Spades | Jack Diamonds | Jack Clubs | 7 Hearts"<<endl;
```

```
break;
}
case'8':{//case for straight flush
cout<<"Straight Flush: Five cards in a sequence, all in the same suit."<<endl;
cout<<"Ex: 4 Spades | 5 Spades | 6 Spades | 7 Spades | 8 Spades"<<endl;
break;
}
case'9':{//case for royal flush
cout<<"Royal Flush: 10, J, Q, K, A (all in the same suit)."<<endl;
cout<<"Ex: 10 Spades | Jack Spades | Queen Spades | King Spades | Ace Spades"<<endl;
break;
}
default:{//default for skip
cho=='N';
skip=true;//set skip to true
break;
}
}
if(skip==false){//if skip is false
cout<<"************************"<<endl;
cout<<"Would you like to view another Hand Ranking?"<<endl;
cout<<"Enter in Y for yes or N for no:"<<endl;
cin>>cho;//prompt for repeat
}
}while(cho=='Y'||cho=='y');//while player wants to repeat
}
//*****************************************//
//* Initialize Deck *//
//*****************************************//
void iniDeck(short cards[],short n){
for(int i=0;i<n;i++){//increment while still in array
cards[i]=i;//for position i, set i to i ex: 41 =41
}
}
//*****************************************//
//* New round *//
//*****************************************//
short round(string z,short x[],short &pl1M,short &pl2M,bool &button){
//Declare variables
short c1=0,c2=0;//player hole cards
short ai1c1=0,ai1c2=0;//computer hole cards
short com1,com2,com3,com4,com5;//community cards
short playCom, aiCom;//number for combination of hand
short pf1=0,pf2=0,pf3=0,pf4=0,pf5=0;//player final hand cards 1-5
short cf1=0,cf2=0,cf3=0,cf4=0,cf5=0;//computer final hand cards 1-5
short usrHand,ai1Hand;//hand values
short hc=0,hc2=0,ai1hc=0,ai1hc2=0;//high card for tie hands
short winNum;//player number who has top hand
string plR,ai1R; //quick string for hand ranking
//variables for betting
short pot=0;//pot total
```

```
bool skip=false;//skip if there is all in
bool playFold=false, comFold=false;
//get ante
cout<<"************************"<<endl;
worth(pl1M,pl2M,pot);
cout<<"New Round"<<endl;
cout<<"Please hit enter to ante up:"<<endl;
cin.get();
cin.ignore();
skip=anteUp(pl1M,pl2M,pot);
if(skip) cout<<"A player is All-In from the Ante"<<endl;
//display totals
cout<<"************************"<<endl;
worth(pl1M,pl2M,pot);
//Deal hole cards
deal1(c1,x);
deal1(c2,x);
deal1(ai1c1,x);
deal1(ai1c2,x);
//display hole cards
cout<<"Your Hole Cards: "<<cardVal(c1)<<"|"<<cardVal(c2)<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
//call betting round 1
cout<<"************************"<<endl;
if(!skip){
skip=bet(pl1M,pl2M,pot,button,playFold,comFold);
}
//Deal 3 Community Cards (Flop)
deal1(com1,x);
deal1(com2,x);
deal1(com3,x);
if(!skip){
cout<<"Your Hole Cards: "<<cardVal(c1)<<"|"<<cardVal(c2)<<endl;
cout<<"Community Cards: "<<cardVal(com1)<<"|"<<cardVal(com2)<<"|"<<cardVal(com3)<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
//betting round 2
skip=bet(pl1M,pl2M,pot,button,playFold,comFold);
}
if(!skip)cout<<"************************"<<endl;
//Deal 1 Community Card (Turn)
deal1(com4,x);
if(!skip){
cout<<"Your Hole Cards: "<<cardVal(c1)<<"|"<<cardVal(c2)<<endl;
cout<<"Community Cards: "<<cardVal(com1)<<"|"<<cardVal(com2)<<"|"<<cardVal(com3)<<
```

```
"|"<<cardVal(com4)<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
skip=bet(pl1M,pl2M,pot,button,playFold,comFold);
}
//Deal 1 Community Card (River)
deal1(com5,x);
//find best combination of cards per player
playCom=bestCom(com1,com2,com3,com4,com5,c1,c2);
aiCom=bestCom(com1,com2,com3,com4,com5,ai1c1,ai1c2);
//set final hand to best combination
finCom(pf1,pf2,pf3,pf4,pf5,com1,com2,com3,com4,com5,c1,c2,playCom);
finCom(cf1,cf2,cf3,cf4,cf5,com1,com2,com3,com4,com5,ai1c1,ai1c2,aiCom);
//high card and sort
hiCard(pf1,pf2,pf3,pf4,pf5);
hiCard(cf1,cf2,cf3,cf4,cf5);
//find hand value of player
usrHand=ranking(pf1,pf2,pf3,pf4,pf5,hc,hc2);
//find hand value of computer
ai1Hand=ranking(cf1,cf2,cf3,cf4,cf5,ai1hc,ai1hc2);
//set string for short hand ranking
plR=sRank(usrHand);
ai1R=sRank(ai1Hand);
//display final community cards
if(!skip)cout<<"************************"<<endl;
cout<<"Your Hole Cards: "<<cardVal(c1)<<"|"<<cardVal(c2)<<endl;
cout<<"Community Cards: "<<cardVal(com1)<<"|"<<cardVal(com2)<<"|"<<cardVal(com3)<<
"|"<<cardVal(com4)<<"|"<<cardVal(com5)<<endl;
cout<<"************************"<<endl;
//state player hand value
if(usrHand==1)cout<<"Your hand value is low with just a high card ("<<cardVal(pf5)<<")."<<endl;
if(usrHand==2)cout<<"Your hand value is average with just a pair."<<endl;
if(usrHand==3)cout<<"You have two pair."<<endl;
if(usrHand==4)cout<<"You have three of a kind. Not too bad."<<endl;
if(usrHand==5)cout<<"Congrats! You have a straight."<<endl;
if(usrHand==6)cout<<"Wow! You have a Flush."<<endl;
if(usrHand==7)cout<<"What a strong hand. You have a Full House!"<<endl;
if(usrHand==8)cout<<"Four of a Kind! Quick, don't let them see your excitement."<<endl;
if(usrHand==9)cout<<"A Straight Flush! You can't be beat!"<<endl;
//Display player hand
cout<<"Your Best Hand: "<<cardVal(pf1)<<"|"<<cardVal(pf2)<<"|"<<cardVal(pf3)<<
"|"<<cardVal(pf4)<<"|"<<cardVal(pf5)<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
if(!skip){
```

```
skip=bet(pl1M,pl2M,pot,button,playFold,comFold);
}
//Unveil all hands
cout<<"************************"<<endl;
worth(pl1M,pl2M,pot);
cout<<" All Player Hands "<<endl;
cout<<"Community Cards: "<<cardVal(com1)<<"|"<<cardVal(com2)<<"|"<<cardVal(com3)<<
"|"<<cardVal(com4)<<"|"<<cardVal(com5)<<endl;
cout<<"**************"<<endl;
cout<<"Your Hole Cards: "<<cardVal(c1)<<"|"<<cardVal(c2)<<endl;
cout<<"Your Hand: "<<cardVal(pf1)<<"|"<<cardVal(pf2)<<"|"<<cardVal(pf3)<<
"|"<<cardVal(pf4)<<"|"<<cardVal(pf5)<<" ("<<plR<<")"<<endl;
cout<<"************************"<<endl;
cout<<"Phil's Hole Cards: "<<cardVal(ai1c1)<<"|"<<cardVal(ai1c2)<<endl;
cout<<"Phil's Hand: "<<cardVal(cf1)<<"|"<<cardVal(cf2)<<"|"<<cardVal(cf3)<<
"|"<<cardVal(cf4)<<"|"<<cardVal(cf5)<<" ("<<ai1R<<")"<<endl;
cout<<"************************"<<endl;
if(!skip)cout<<"Please hit enter to find the winning hand:"<<endl;
if(skip)cout<<"Please hit enter to continue"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
//find winning hand
if(playFold)winNum=2;
if(comFold)winNum=1;
if(playFold==false&&comFold==false){
winNum=winHand(usrHand,ai1Hand,hc,ai1hc,hc2,ai1hc2);//find winNum
}
string rndWinr=winner(z,winNum);//find winner name
if(playFold)cout<<"Phil wins since you folded"<<endl;
else if(comFold)cout<<"You win since Phil folded"<<endl;
else cout<<rndWinr<<" has the winning hand. "<<endl;//state winner
cout<<rndWinr<<" wins the pot: $"<<pot<<endl;
//award pot to winner
if(winNum==1) award(pl1M,pot);
if(winNum==2) award(pl2M,pot);
if(winNum<1||winNum>2)split(pl1M,pl2M,pot);
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
//display player totals
cout<<"Player Totals:"<<endl;
cout<<"You: $"<<pl1M<<" | Phil: $"<<pl2M<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
//increment button
if(button==false) button=true;
else if(button==true) button=false;
```

```
return winNum;//return winner number
}
//*******************************************//
//* Deal 1 *//
//*******************************************//
void deal1(short &c1,short y[]){
//declare variables
char repeat;//repeat
short card;//new card
do{//do
card=rand()%52+1;//pick random number
if(y[card]==0){//if card is 0, then its already picked, so repeat
repeat='Y';
}else{//else dont repeat
repeat='N';
}
}while(repeat=='Y');//while need new card
y[card]=0;//set the card you picked to zero in array
c1=card;//c1 = card
}
//*******************************************//
//* Card Values *//
//*******************************************//
string cardVal(short v){
string f;//declare string f for card name
//Spades
if(v==1)f="2 Spades";
if(v==2) f="3 Spades";
if(v==3) f="4 Spades";
if(v==4) f="5 Spades";
if(v==5) f="6 Spades";
if(v==6) f="7 Spades";
if(v==7) f="8 Spades";
if(v==8) f="9 Spades";
if(v==9) f="10 Spades";
if(v==10) f="Jack Spades";
if(v==11) f="Queen Spades";
if(v==12) f="King Spades";
if(v==13) f="Ace Spades";
//Hearts
if(v==14) f="2 Hearts";
if(v==15) f="3 Hearts";
if(v==16) f="4 Hearts";
if(v==17) f="5 Hearts";
if(v==18) f="6 Hearts";
if(v==19) f="7 Hearts";
if(v==20) f="8 Hearts";
if(v==21) f="9 Hearts";
if(v==22) f="10 Hearts";
if(v==23) f="Jack Hearts";
if(v==24) f="Queen Hearts";
```

```
if(v==25) f="King Hearts";
if(v==26) f="Ace Hearts";
//Clubs
if(v==27) f="2 Clubs";
if(v==28) f="3 Clubs";
if(v==29) f="4 Clubs";
if(v==30) f="5 Clubs";
if(v==31) f="6 Clubs";
if(v==32) f="7 Clubs";
if(v==33) f="8 Clubs";
if(v==34) f="9 Clubs";
if(v==35) f="10 Clubs";
if(v==36) f="Jack Clubs";
if(v==37) f="Queen Clubs";
if(v==38) f="King Clubs";
if(v==39) f="Ace Clubs";
//Diamonds
if(v==40) f="2 Diamonds";
if(v==41) f="3 Diamonds";
if(v==42) f="4 Diamonds";
if(v==43) f="5 Diamonds";
if(v==44) f="6 Diamonds";
if(v==45) f="7 Diamonds";
if(v==46) f="8 Diamonds";
if(v==47) f="9 Diamonds";
if(v==48) f="10 Diamonds";
if(v==49) f="Jack Diamonds";
if(v==50) f="Queen Diamonds";
if(v==51) f="King Diamonds";
if(v==52) f="Ace Diamonds";
return f;//return card name
}
//*******************************************//
//* Hand Ranking *//
//*******************************************//
short ranking(short a,short b,short c,short d,short e, short &hi1, short &hi2){
//declare variables
//high card hand value= 1
short value=1,temp;//temp
char str='N',flu='N';// straight = n, flush = n
//flush = 6
if((a>=1&&a<=13)&&(b>=1&&b<=13)&&(c>=1&&c<=13)&&(d>=1&&d<=13)&&(e>=1&&e<=13)) value=6,flu='Y'; //spades
if((a>=14&&a<=26)&&(b>=14&&b<=26)&&(c>=14&&c<=26)&&(d>=14&&d<=26)&&(e>=14&&e<=26)) value=6,flu='Y';
//hearts
if((a>=27&&a<=39)&&(b>=27&&b<=39)&&(c>=27&&c<=39)&&(d>=27&&d<=39)&&(e>=27&&e<=39)) value=6,flu='Y';
//clubs
if((a>=40&&a<=52)&&(b>=40&&b<=52)&&(c>=40&&c<=52)&&(d>=40&&d<=52)&&(e>=40&&e<=52)) value=6,flu='Y';
//diamonds
//get rid of suits
for(a;a>13;a-=13);
for(b;b>13;b-=13);
for(c;c>13;c-=13);
```

```
for(d;d>13;d-=13);
for(e;e>13;e-=13);
//sort for straight
//sort first value
if(a>b) temp=a,a=b,b=temp;
if(a>c) temp=a,a=c,c=temp;
if(a>d) temp=a,a=d,d=temp;
if(a>e) temp=a,a=e,e=temp;
//sort second value
if(b>c) temp=b,b=c,c=temp;
if(b>d) temp=b,b=d,d=temp;
if(b>e) temp=b,b=e,e=temp;
//sort third value
if(c>d) temp=c,c=d,d=temp;
if(c>e) temp=c,c=e,e=temp;
//sort fourth value
if(d>e) temp=d,d=e,e=temp;
//high card for flush
if(value==6) hi1=e,hi2=d;
//high card for high card hand
if(value==1)hi1=e,hi2=d;
//pair = 2
if(a==b||a==c||a==d) value=2,hi1=a,hi2=e;
if(a==e)value=2,hi1=a,hi2=d;
if(b==c||b==d) value=2,hi1=b,hi2=e;
if(c==d) value=2,hi1=c,hi2=e;
if(c==e) value=2,hi1=c,hi2=d;
if(d==e) value=2,hi1=e,hi2=c;
if(b==e) value=2,hi1=e,hi2=d;
//two pair = 3
if((b==c&&d==e)||(b==e&&c==d)) value=3,hi1=d,hi2=b;
if((b==d&&c==e)||(a==c&&b==e)) value=3, hi1=c, hi2=e;
if((a==e&&c==d)||(a==d&&c==e)) value=3,hi1=c,hi2=a;
if((a==c&&d==e)||(a==e&&b==d)) value=3, hi1=d, hi2=a;
if((a==b&&d==e)||(a==d&&b==e)) value=3,hi1=e, hi2=a;
if((a==b&&c==e)||(a==e&&b==c)) value=3,hi1=c,hi2=a;
if((a==b&&c==d)||(a==c&&b==d)) value=3,hi1=d,hi2=a;
if((a==d&&b==c)) value=3, hi1=c,hi2=a;
//three of a kind = 4
if((a==d&&a==e)) value=4,hi1=a,hi2=c;
if((a==b&&a==e)||(a==c&&a==e)) value=4, hi1=a, hi2=d;
if((a==b&&a==c)||(a==b&&a==d)||(a==c&&a==d)) value=4, hi1=a,hi2=e;
if((c==d&&c==e)) value=4,hi1=e, hi2=b;
if((b==d&&b==e)) value=4, hi1=e, hi2=c;
if((b==c&&b==e)) value=4,hi1=c, hi2=d;
if(b==c&&b==d) value=4,hi1=b,hi2=e;
//straight = 5
if((e-d==1)&&(d-c==1)&&(c-b==1)&&(b-a==1)) value=5,str='Y',hi1=e,hi2=d;
if((a==1)&&(b==2)&&(c==3)&&(d==4)&&(e==13)) value=5,str='Y',hi1=e,hi2=d;
//full house = 7
if((c==d&&c==e&&a==b)||(a==d&&a==e&&b==c)) value=7,hi1=d,hi2=b;
```

```
if((a==b&&a==e&&c==d)||(a==b&&a==c&&d==e)) value=7,hi1=a,hi2=d;
if((a==c&&a==e&&b==d)||(b==c&&b==e&&a==d)) value=7,hi1=c,hi2=d;
if((a==b&&a==d&&c==e)||(a==c&&a==d&&b==e)) value=7, hi1=a, hi2=e;
if((b==d&&b==e&&a==c)||(b==c&&b==d&&a==e)) value=7,hi1=b,hi2=a;
//four of a kind = 8
if((b==c&&d==e&&c==d)||(a==c&&d==e&&c==d)||(a==b&&d==e&&b==d)) value=8,hi1=d,hi2=d;
if((a==b&&c==e&&b==c)||(a==b&&c==d&&b==c)) value=8,hi1=a,hi2=d;
//straight flush = 9
if(flu=='Y'&&str=='Y') value=9;// if flush and straight then value =9
return value;//return hand value
}
//*********************************************//
//* Short Hand Rank *//
//*********************************************//
string sRank(short a){
string state;//declare string for hand type
if(a==1)state="High card";//if 1 then high card
if(a==2)state="Pair";//if 2 then pair
if(a==3)state="Two Pair";//if 3 then two pair
if(a==4)state="Three of a Kind";// if 4 then 3 of a kind
if(a==5)state="Straight";//if 5 then straight
if(a==6)state="Flush";//if 6 then flush
if(a==7)state="Full House";// 7 then full house
if(a==8)state="Four of a Kind";//if 8 then four of a kind
if(a==9)state="Straight Flush";// if 9 then straight flush
return state;//return string for hand type
}
//*********************************************//
//* High Card/Sort *//
//*********************************************//
void hiCard(short &a,short &b,short &c,short &d,short &e){
short tempa=a,tempb=b,tempc=c,tempd=d,tempe=e;//temp variables for swap
short temp=0;
//get rid of suits
for(tempa;tempa>13;tempa-=13);//take away 13 for values above 13
for(tempb;tempb>13;tempb-=13);
for(tempc;tempc>13;tempc-=13);
for(tempd;tempd>13;tempd-=13);
for(tempe;tempe>13;tempe-=13);
//sort first value
if(tempa>tempb){
temp=tempa,tempa=tempb,tempb=temp;
temp=a,a=b,b=temp;//swap if needed
}
if(tempa>tempc){
temp=tempa,tempa=tempc,tempc=temp;
temp=a,a=c,c=temp;
}
if(tempa>tempd){
temp=tempa,tempa=tempd,tempd=temp;
temp=a,a=d,d=temp;
```

```
}
if(tempa>tempe){
temp=tempa,tempa=tempe,tempe=temp;
temp=a,a=e,e=temp;
}
//sort second value
if(tempb>tempc){//swap if needed
temp=tempb,tempb=tempc,tempc=temp;
temp=b,b=c,c=temp;
}
if(tempb>tempd){
temp=tempb,tempb=tempd,tempd=temp;
temp=b,b=d,d=temp;
}
if(tempb>tempe){
temp=tempb,tempb=tempe,tempe=temp;
temp=b,b=e,e=temp;
}
//sort third value
if(tempc>tempd){
temp=tempc,tempc=tempd,tempd=temp;
temp=c,c=d,d=temp;//swap if needed
}
if(tempc>tempe){
temp=tempc,tempc=tempe,tempe=temp;
temp=c,c=e,e=temp;
}
//sort fourth value
if(tempd>tempe){
temp=tempd,tempd=tempe,tempe=temp;
temp=d,d=e,e=temp;//swap if needed
}
}
//*****************************************//
//* Best Combination *//
//*****************************************//
short bestCom(short a,short b,short c,short d,short e,short h1,short h2){
//variables
short high,temp, placeH,placeH2,bestCom=0,curHC=0,curHC2;//high,temporary,placeholder,best combination
number,current High Card
high=ranking(a,b,c,d,e,curHC,curHC2);//5 community cards
//3 com cards and 2 hole
temp=ranking(h1,h2,a,b,c,placeH,placeH2);
if(temp>high) bestCom=1, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=1,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=1,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,a,c,d,placeH,placeH2);
```

```
if(temp>high) bestCom=2, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=2,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=2,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,a,d,e,placeH,placeH2);
if(temp>high) bestCom=3, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=3,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=3,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,a,b,d,placeH,placeH2);
if(temp>high) bestCom=4, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=4,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=4,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,a,b,e,placeH,placeH2);
if(temp>high) bestCom=5, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=5,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=5,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,a,c,e,placeH,placeH2);
if(temp>high) bestCom=6, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=6,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=6,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,b,c,d,placeH,placeH2);
if(temp>high) bestCom=7, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=7,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=7,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,b,c,e,placeH,placeH2);
if(temp>high) bestCom=8, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=8,high=temp,curHC=placeH,curHC2=placeH2;
```

```
if(placeH==curHC){
if(placeH2>curHC)bestCom=8,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,b,d,e,placeH,placeH2);
if(temp>high) bestCom=9, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=9,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=9,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,h2,c,d,e,placeH,placeH2);
if(temp>high) bestCom=10, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=10,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=10,high=temp,curHC=placeH,curHC2=placeH2;
}
}
//4 com card and first hole card
temp=ranking(h1,a,b,c,d,placeH,placeH2);
if(temp>high) bestCom=11, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=11,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=11,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,a,b,c,e,placeH,placeH2);
if(temp>high) bestCom=12, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=12,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=12,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,a,c,d,e,placeH,placeH2);
if(temp>high) bestCom=13, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=13,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=13,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h1,a,b,d,e,placeH,placeH2);
if(temp>high) bestCom=14, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=14,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=14,high=temp,curHC=placeH,curHC2=placeH2;
```

```
}
}
temp=ranking(h1,b,c,d,e,placeH,placeH2);
if(temp>high) bestCom=15, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=15,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=15,high=temp,curHC=placeH,curHC2=placeH2;
}
}
//4 com card and second hole card
temp=ranking(h2,a,b,c,d,placeH,placeH2);
if(temp>high) bestCom=16, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=16,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=16,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h2,a,b,c,e,placeH,placeH2);
if(temp>high) bestCom=17, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=17,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=17,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h2,a,c,d,e,placeH,placeH2);
if(temp>high) bestCom=18, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=18,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=18,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h2,a,b,d,e,placeH,placeH2);
if(temp>high) bestCom=19, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=19,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=19,high=temp,curHC=placeH,curHC2=placeH2;
}
}
temp=ranking(h2,b,c,d,e,placeH,placeH2);
if(temp>high) bestCom=20, high=temp;
if(temp==high){
if(placeH>curHC)bestCom=20,high=temp,curHC=placeH,curHC2=placeH2;
if(placeH==curHC){
if(placeH2>curHC)bestCom=20,high=temp,curHC=placeH,curHC2=placeH2;
}
}
```

```
return bestCom;
}
//*******************************************//
//* Set Final Hand *//
//*******************************************//
void finCom(short &c1,short &c2,short &c3, short &c4,short &c5,short a,short b,short c,short d,short
e,short h1,short h2,short comb){
if(comb==0) c1=a,c2=b,c3=c,c4=d,c5=e;
if(comb==1) c1=h1,c2=h2,c3=a,c4=b,c5=c;
if(comb==2) c1=h1,c2=h2,c3=a,c4=c,c5=d;
if(comb==3) c1=h1,c2=h2,c3=a,c4=d,c5=e;
if(comb==4) c1=h1,c2=h2,c3=a,c4=b,c5=d;
if(comb==5) c1=h1,c2=h2,c3=a,c4=b,c5=e;
if(comb==6) c1=h1,c2=h2,c3=a,c4=c,c5=e;
if(comb==7) c1=h1,c2=h2,c3=b,c4=c,c5=d;
if(comb==8) c1=h1,c2=h2,c3=b,c4=c,c5=e;
if(comb==9) c1=h1,c2=h2,c3=b,c4=d,c5=e;
if(comb==10) c1=h1,c2=h2,c3=c,c4=d,c5=e;
if(comb==11) c1=h1,c2=a,c3=b,c4=c,c5=d;
if(comb==12) c1=h1,c2=a,c3=b,c4=c,c5=e;
if(comb==13) c1=h1,c2=a,c3=c,c4=d,c5=e;
if(comb==14) c1=h1,c2=a,c3=b,c4=d,c5=e;
if(comb==15) c1=h1,c2=b,c3=c,c4=d,c5=e;
if(comb==16) c1=h2,c2=a,c3=b,c4=c,c5=d;
if(comb==17) c1=h2,c2=a,c3=b,c4=c,c5=e;
if(comb==18) c1=h2,c2=a,c3=c,c4=d,c5=e;
if(comb==19) c1=h2,c2=a,c3=b,c4=d,c5=e;
if(comb==20) c1=h2,c2=b,c3=c,c4=d,c5=e;
}
//*******************************************//
//* Winning Hand *//
//*******************************************//
short winHand(short a,short b,short h1,short h2,short sh1, short sh2){
//find top hand value
short topHand=a,pl1=0,pl2=0,win;//declare variables for ties
if(b>topHand) topHand=b;
//find all that have the same value of the top hand
if(a==topHand) pl1=1;
if(b==topHand) pl2=1;
//compare ties for top hand
if((pl1==1)&&(pl2==1)){//if two have tie
if(h1>h2) win=1;//win =1
if(h2>h1) win=2;//win =2
if(h1==h2){
if(sh1>sh2) win=1;
if(sh2>sh1) win=2;
}
}
//if no tie
if((pl1==1)&&(pl2==0)) win=1;//win 1
if((pl1==0)&&(pl2==1)) win=2;//win 2
```

```
//return win
return win;
}
//*****************************************//
//* Winning Player *//
//*****************************************//
string winner(string you,short a){
string winner;//declare string variable for winner
if(a==1) winner=you;//if 1 then winner = user
if(a==2) winner="Phil";//if 2 then winner = tom
if(a<1||a>2) winner="(Tie Game) No one";// else tie game
return winner;//return winner
}
//*****************************************//
//* Worth *//
//*****************************************//
void worth(short a,short b,short p){
cout<<"You:$"<<a<<" | Phil:$"<<b<<" | Pot:$"<<p<<endl;
}
//*****************************************//
//* AnteUp *//
//*****************************************//
bool anteUp(short &a, short &b,short &pot){
//variables
bool allIn=false,skip=false;
//call indAnte
allIn=indAnte(a,pot);//player 1 ante
if(allIn==true) skip=true;
allIn=indAnte(b,pot);//player 2 ante
if(allIn==true) skip=true;
//find if skip is needed
return skip;
}
//*****************************************//
//* Individual Ante *//
//*****************************************//
bool indAnte(short &a, short &p){
bool allIn, fold=false;
if(a==0) fold=true;
if(a>0) a-=5, p+=5;
if(a==0&&fold==false) allIn=true;
return allIn;
}
//*****************************************//
//* Betting Function *//
//*****************************************//
bool bet(short &pM,short &cM, short &pot,bool button,bool &playFold,bool &comFold){
//declare variables
bool stop=false,skip=false;
short tally=0;//tally to make sure both players bet at least once
short pB=0,cB=0,curBet=0;//player/computer total bet and current bet
```

```
short allFold;
do{
if(button==true){
//call player bet
if(!stop&&!comFold){
if(tally<1)cout<<"You bet first:"<<endl;
playBet(pM,pB,curBet,pot,playFold,cM);
}
tally++;
if(tally>1)stop=check(pB,cB,curBet,playFold,comFold,pM,cM);
}
//call computer bet
if(!stop&&!playFold){
if(tally<1)cout<<"Phil bets first:"<<endl;
compBet(cM,cB,curBet,pot,comFold,pM);
tally++;
if(tally>1)stop=check(pB,cB,curBet,playFold,comFold,pM,cM);
}
if(button==false){
//call player bet
if(!stop&&!comFold){
playBet(pM,pB,curBet,pot,playFold,cM);
tally++;
stop=check(pB,cB,curBet,playFold,comFold,pM,cM);
}
}
//check for raises
stop=check(pB,cB,curBet,playFold,comFold,pM,cM);
}while(stop==false);
skip=cease(pM,cM,playFold,comFold);
worth(pM,cM,pot);
return skip;
}
//*****************************************//
//* Player Bet *//
//*****************************************//
void playBet(short &pM,short &pB,short &curBet,short &pot,bool &playFold,short cM){
//variables
short cho;
//display current bet
worth(pM,cM,pot);
if(pB>0)cout<<"Your previous bet: $"<<pB<<endl;
cout<<"Current bet to match: $"<<curBet<<endl;
cout<<"Pot:$"<<pot<<endl;
//prompt for bet
do{
if(curBet==0)cout<<"Enter in 1-check $"<<curBet<<":"<<endl;
if(curBet>0)cout<<"Enter in 1-call $"<<curBet<<":"<<endl;
if((curBet>0)&&(cM>0)&&(pM>5))cout<<" 2-re-raise $"<<curBet+5<<":"<<endl;
if((curBet==0)&&(cM>0))cout<<" 2-bet $"<<curBet+5<<":"<<endl;
cout<<" 3-fold:"<<endl;
```

```
cin>>cho;
}while(cho<1||cho>3);
if(cho==1){
if((curBet-pB)>pM)pot+=pM,pM-=pM;
else pM-=(curBet-pB),pot+=(curBet-pB),pB=curBet;
}
if(cho==2){
if((curBet-pB)>pM)pot+=pM,pM-=pM;
curBet+=5,pM-=(curBet-pB),pot+=(curBet-pB),pB=curBet;
}
if(cho==3){
playFold=true;
}
if(pM==0){
cout<<"You are All-In!"<<endl;
worth(pM,cM,pot);
}
}
//*********************************************//
//* Computer Bet *//
//*********************************************//
void compBet(short &cM,short &cB,short &curBet,short &pot,bool &comFold,short pM){
//variables
short cho;
cout<<"************************"<<endl;
worth(pM,cM,pot);
if(cB>0)cout<<"Phil's previous bet $"<<cB<<endl;
cout<<"The current bet to match is $"<<curBet<<endl;
cout<<"Pot:$"<<pot<<endl;
cout<<"It is Phil's turn to bet:"<<endl;
if(curBet==0)cho=rand()%8+1;
else if((curBet-cB)>=cM)cho=1;
else cho=rand()%10+1;
if(cho<=5){
if((curBet-cB)>cM)pot+=cM,cM-=cM;
else cM-=(curBet-cB),pot+=(curBet-cB),cB=curBet;
if(curBet==0) cout<<"Phil checks."<<endl;
if(curBet>0) cout<<"Phil calls the bet."<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
}
if(cho>5&&cho<=8){
if(curBet==0)cout<<"Phil bets $5."<<endl;
if(curBet>0)cout<<"Phil re-raises to $";
curBet+=5,cM-=(curBet-cB),pot+=(curBet-cB),cB=curBet;
cout<<curBet<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
```

```
cin.get();
cin.ignore();
cout<<"************************"<<endl;
}
if(cho>8){
cout<<"Phil folds."<<endl;
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
comFold=true;
}
if(cM==0){
curBet==cB;
cout<<"Phil is All-In"<<endl;
worth(pM,cM,pot);
cout<<"************************"<<endl;
cout<<"Please hit enter to continue:"<<endl;
cin.get();
cin.ignore();
cout<<"************************"<<endl;
}
}
//*****************************************//
//* Check for Bet end *//
//*****************************************//
bool check(short pB,short cB,short curBet,bool &playFold,bool &comFold,short pM,short cM){
if(pB==curBet&&cB==curBet)return true;
if(playFold==true||comFold==true)return true;
if(cM==0&&pB>=curBet)return true;
if(pM==0&&cB>=curBet)return true;
return false;
}
//*****************************************//
//* Award pot to winner *//
//*****************************************//
void award(short &w,short pot){
w+=pot;
}
//*****************************************//
//* Split the pot *//
//*****************************************//
void split(short &a,short &b,short &pot){
short temp=pot/2;
a+=temp;
b+=temp;
}
//*****************************************//
//* All in or Fold *//
//*****************************************//
```

```
bool cease(short pM,short cM,bool playFold,bool comFold){
if((pM==0)||(cM==0)||(playFold==true)||comFold==true)return true;
else return false;
}
//Fill 2D array
void loose(char a[][2],bool){
a[0][0]='1',a[0][1]='2';
a[1][0]='3',a[1][1]='4';
}
```