Project 1

Blackjack

Christian Villanueva
CSC 17A - 49213
25 October 2020

Introduction:

I have created a program that runs a simple version of the card game Blackjack from scratch. This version of Blackjack is the card aspect only. I have not yet implemented a chip system for betting, nor have I included the ability to split when a pair is drawn. These will come in the next iteration of the program. Blackjack is my favorite card game, and it seemed fitting to make, given the requirements for the project, and my prerequisite knowledge of the game's rules. The entire program was created from the ground up, and no reference code was used throughout the entirety of the project.

The player is dealt two cards and one of the dealer cards is shown. The player is then given the choice to "hit" or "stay". If the player chooses to hit, then the player is given another card. If the player's card total (each card is given a number value, and is added up to find the total) hits or exceeds 21, they automatically stay. The dealer then reveals their second card. After the dealer reveals their second card, if their total is less than or equal to sixteen, they draw until their total exceeds sixteen. The card total of the dealer and the player are compared, and whichever total is closer to 21 wins, given that the total does not exceed 21 (if the player or dealer exceeds a total of 21, they lose the game). If the difference from 21 is the same or if both the dealer and player exceed 21 or "bust" then the game results in a draw. If the player chooses to stay, then their total remains, and the dealer continues the same as previously described.

Summary:

Total Lines: 468

Blank Lines/Comments Only: 68

Lines of Code/Code With Comments: 373

Header File Lines: 27

Number of Structures: 2

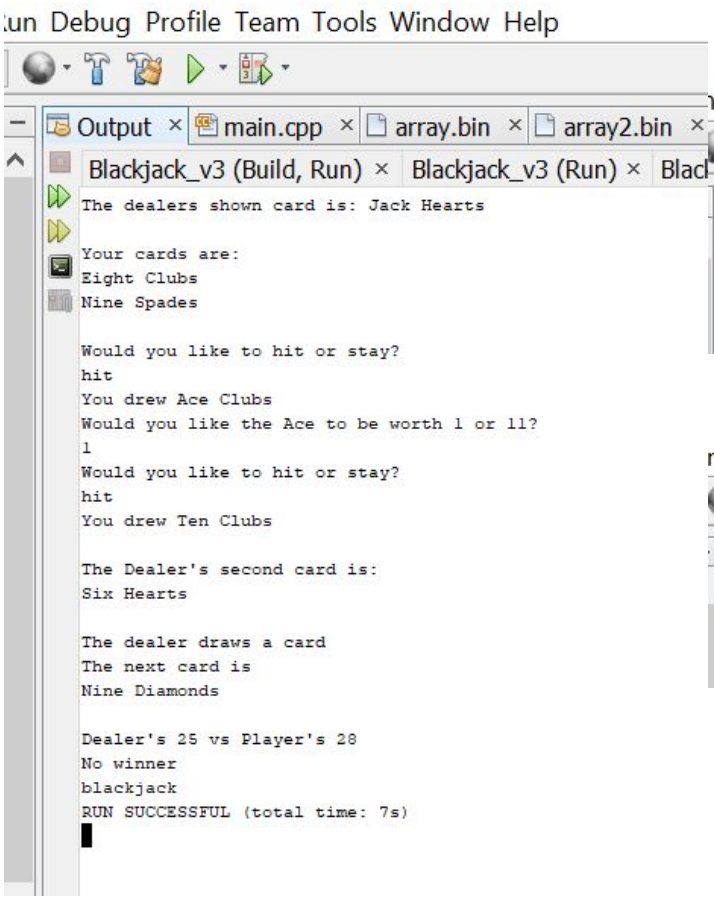
Number of Variables: 28

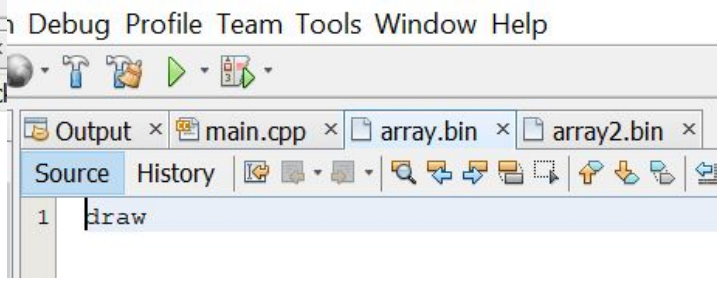
The project took me the two weeks to complete. Roughly 30 hours were put into the project as a whole including the documentation. The most challenging aspect of the project was implementing all of the required concepts into the program. The game alone form was roughly 330 lines of code before revision to include the concepts. The most difficult concept to add was binary file operation. This took up the last 5 days of the project along with other minor improvements to the code for enhanced readability, and reduction of the main function size.

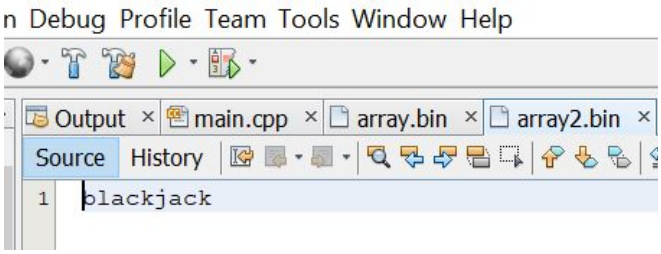
Description:

The game initially started as just a program that drew cards and output them. The card values and the ability to total them were then added, as well as the conditions for winning the game. Next came checks to ensure that repeat cards did not show up. After this came the rest of the concepts from the class up until this point such as pointers and binary files.

Sample output:







Player Input:

“hit”, “1”, “hit”,

Debug Profile Team Tools Window Help

Output × main.cpp × array.bin × array2.bin ×

Blackjack_v3 (Build, Run) × Blackjack_v3 (Run) × Blackj

```
The dealers shown card is: King Hearts

Your cards are:
Ten Hearts
Eight Hearts

Would you like to hit or stay?
hit
You drew Two Hearts
Would you like to hit or stay?
stay

The Dealer's second card is:
Four Clubs

The dealer draws a card
The next card is
Ace Hearts

The dealer draws a card
The next card is
Six Clubs

Dealer's 21 vs Player's 20
You lose
blackjack
RUN SUCCESSFUL (total time: 7s)
```

Debug Profile Team Tools Window Help

Output × main.cpp × array.bin × array2.bin ×

Source History

```
1  loser
```

Debug Profile Team Tools Window Help

Output × main.cpp × array.bin × array2.bin ×

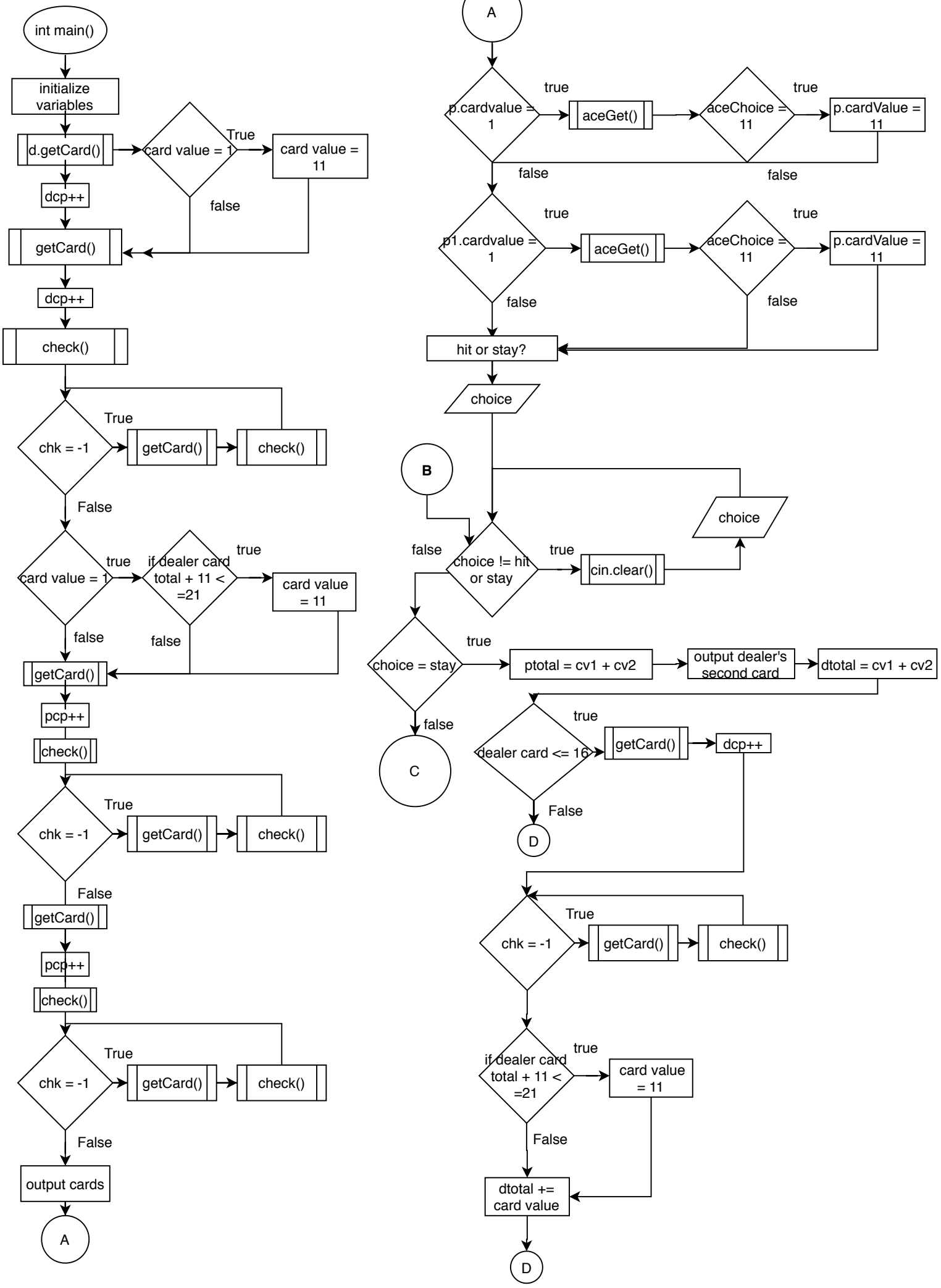
Source History

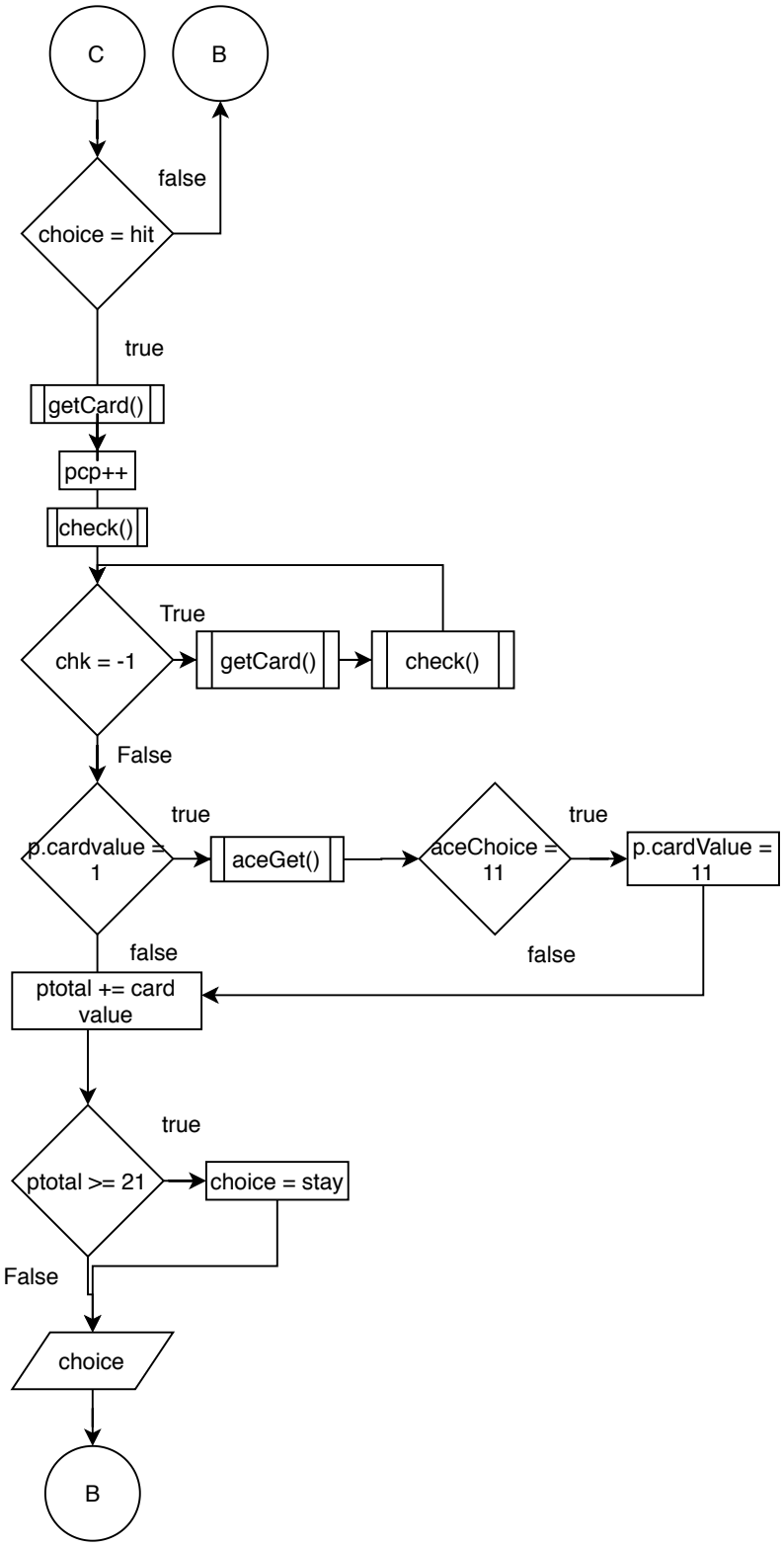
```
1  blackjack
```

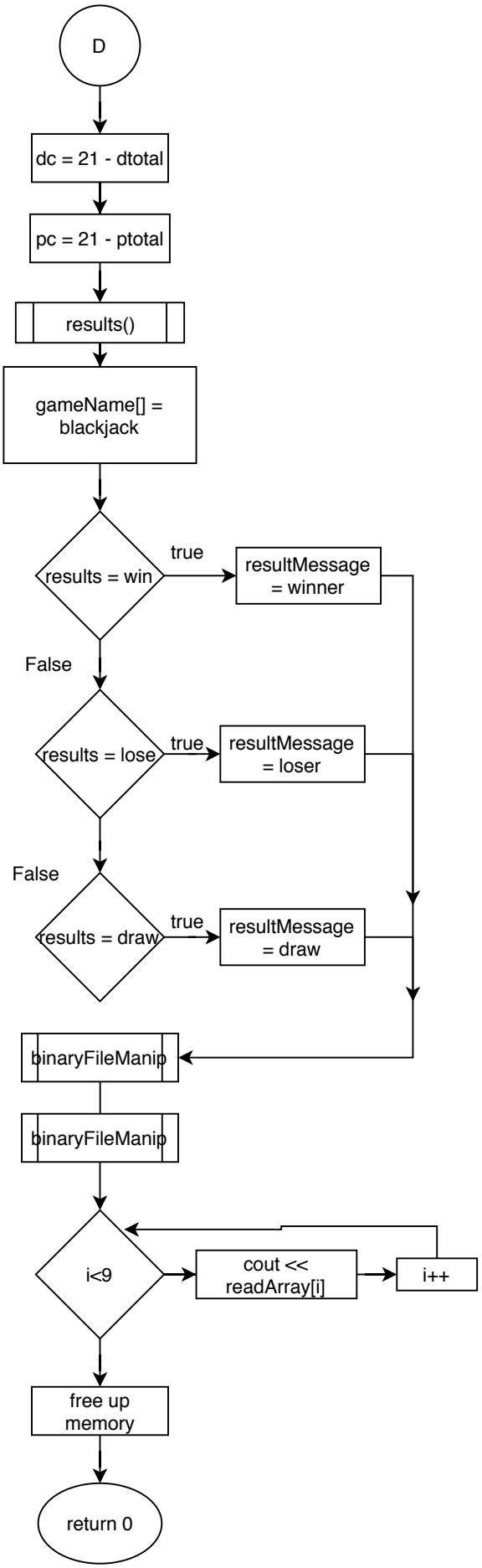
Player Input:

“hit”, “stay”

Note: I use abbreviations for variable names




C
B
choice = hit
false
true
getCard()
pcp++
check()
chk = -1
True
getCard()
check()
False
p.cardvalue = 1
true
aceGet()
aceChoice = 11
true
p.cardValue = 11
false
false
ptotal += card value
ptotal >= 21
true
choice = stay
False
choice
B


D
dc = 21 - dtotal
pc = 21 - ptotal
results()
gameName[] = blackjack
results = win
true
resultMessage = winner
False
results = lose
true
resultMessage = loser
False
results = draw
true
resultMessage = draw
binaryFileManip
binaryFileManip
i<9
cout << readArray[i]
i++
free up memory
return 0

Version 5 Pseudocode

```
Enum gameResult = lose, win, draw

Main
{
Set random number seed
define 10 structures each to hold player and dealer cards
Define structure to hold win results
Define variables to hold card value totals, cards in play, score, checks, and choices

Draw dealer cards
Increment number of cards in play
Check for/replace  repeat cards
If an ace is drawn, make its value 11 if it doesn't bust the dealer

Draw player cards
Increment number of cards in play
Check for/replace repeat cards

Output dealer's first card and player's two cards
If the player draws a(n) ace(s) gets their choice for the aces value(only allows 1 or 11)
Gets the player's choice to hit or stay with input validation

If stay is chosen,
        Calculates player card total
        Reveals dealer's second card
        If dealer's card total is <= 16,
                draws cards, iterates cards in play, and checks
                for/replaces repeats until a total of >16 is achieved
                If dealer draws ace, sets it to 11 if it doesn't bust the dealer
        Calculates dealer card total
```

```
If hit is chosen, loops until stay is chosen
        Calculates player card total
        Draw card and iterate cards in play
        Checks for/replaces repeat cards

        Display card drawn
        If ace is drawn, get choice for ace value(only allows 1 or 11)
        Add card value to player card total

        If player card total >= 21, set choice to stay
        If player card total < 21, get choice to hit or stay(with input validation)

        Reveals dealer's second card
        If dealer's card total is <= 16,
                draws cards, iterates cards in play, and checks
                for/replaces repeats until a total of >16 is achieved
                If dealer draws ace, sets it to 11 if it doesn't bust the dealer
        Calculates dealer card total

Find difference of dealer and player total from 21
Find and output game results based on difference

Define character arrays to hold  result message and game name
If win result is win, message is winner
If win result is lose, message is loser
If win result is draw, message is draw

Write the result message to a binary file
Write the game name to another binary file
Write the first card information into another binary file

Output the game name

Free up the used memory and exit the program

}
```

```
getCard
{
Get random number between 1 and 13
Get random number between 1 and 4
Assign card name and value to first number drawn
1 = ace, with value 1
2 = two, with value 2
3 = three, with value 3
4 = four, with value 4
5 = five, with value 5
6 = six, with value 6
7 = seven, with value 7
8 = eight, with value 8
9 = nine, with value 9
10 = ten, with value 10
11 = jack, with value 10
12 = queen, with value 10
13 = king, with value 10

Assign card suit to second number drawn
1 = spades
2 = clubs
3 = diamonds
4 = hearts

}
```

```
check
{
Compare card passed in with every card played before it
If card passed has been played before, return -1
Otherwise return 1
}

aceGet
{
Prompt user to choose value 1 or 11 for ace
If neither 1 or 11 is entered, output error message and prompt another input
Return the value chosen
}

binaryFileManip
{
Define pointer to hold a char array
Create file stream for input and output to a binary file
Go to start of file
Write the passed array to the file
Go back to start of file
Read the char array from the file
Close the file

}
```

results
{
define structure to return
Define lose message to display

Output dealer and player totals
If both dealer and player total difference are 0, output no winner and the result is recorded as draw
If dealer and player total differences are the same, output no winner and the result is recorded as draw
If both player and dealer total difference is negative, output no winner and the result is recorded as draw
If dealer total is negative and player total isn't, output win message and the result is recorded as win
If player total is negative and the dealer total isn't, output lose message and the result is recorded as lose
If dealer's total is smaller than player's and is not negative, output lose message and the result is recorded as lose
If player's total is smaller than the dealer's and is not negative, output win message and the result is recorded as win

Return the structure holding the results
}

binaryRecord
{
Create file stream to write to a binary file
Go to the start of the file
Write the passed structure to the file
Close the file
}

Variable Type	Name	Line
int	winc	10(winTracker.h)
	cardValue	9(card.h)
	dtotal	29
	ptotal	30
	dc	31
	pc	32
	chk	33
	dcp	34
	pcp	35
	aceChoice	36
	dh	121
	ph	149
	n	283
	s	284
	ac	355
string	cardName	8(card.h)
	suit	10(card.h)
	choice	28
char	resultMessage[7]	233
	gameName[9]	234
	*readArray1	236
	*readArray2	237
	result[]	373
	*arr	421
enum	gameResult	11
card	player[10]	24
	dealer[10]	25
winTracker	gameResults	27
	s	372

CSC/CIS 17A Project 1 Check-Off Sheet

Chapter	Section	Concept	Points for Inclusion	Location in Code	Comments
9		**Pointers/Memory Allocation**			
	1	Memory Addresses			
	2	Pointer Variables	5	421	
	3	Arrays/Pointers	5	421	
	4	Pointer Arithmetic			
	5	Pointer Initialization			
	6	Comparing			
	7	Function Parameters	5	419	
	8	Memory Allocation	5	422	
	9	Return Parameters	5	431	
	10	Smart Pointers			
10		**Char Arrays and Strings**			
	1	Testing			
	2	Case Conversion			
	3	C-Strings	10	373	
	4	Library Functions			
	5	Conversion			
	6	Your own functions			
	7	Strings	10	104	
11		**Structured Data**			
	1	Abstract Data Types			
	2	Data			
	3	Access			
	4	Initialize			
	5	Arrays	5	24	
	6	Nested	5	408	
	7	Function Arguments	5	341	
	8	Function Return	5	416	
	9	Pointers	5	26	
	10	Unions ****			
	11	Enumeration	5	11	
12		**Binary Files**			
	1	File Operations			
	2	Formatting	2	423	
	3	Function Parameters	2		
	4	Error Testing			
	5	Member Functions	2	424	
	6	Multiple Files	2	266	
	7	Binary Files	5	423	
	8	Records with Structures	5	438	
	9	Random Access Files	5	425	
	10	Input/Output Simultaneous	2	423	
		Total	100		

Code:

card.h:

```
//holds individual cards


#ifndef CARD_H
#define CARD_H


struct card
{
    std::string cardName;
    int cardValue;
    std::string suit;
};


#endif /* CARD_H */
```

winTracker.h:

```
//keeps track of win condition


#ifndef WINTRACKER_H

#define WINTRACKER_H


struct winTracker

{

    struct op

    {

        int winc;

    } wr;

};


#endif /* WINTRACKER_H */
```

main.cpp:

```
#include <iostream>

#include <ctime>

#include <cstring>

#include <fstream>

#include "card.h"

#include "winTracker.h"

using namespace std;



//game can either end in win, loss, or a draw

enum gameResult{lose, win, draw};


void getCard(struct card &); //draws a card

int check(struct card[], struct card[], struct card, int, int); //checks if card has been

already been played

int aceGet(); //gets the value of ace that the player decides

struct winTracker results(int, int, int, int); //outputs game results and records the result

char *binaryFileManip(string, char *, int); //reads and writes to binary files

void binaryRecord(struct card); //writes a card structure to a binary file



int main()
```

```
{
    srand(time(0)); //random number seed
    struct card player[10]; //player cards
    struct card dealer[10]; //dealer cards
    struct winTracker *gameResults; //pointer to structure
    gameResults = new winTracker;
    string choice; //stores hit or stay
    int dtotal = 0; //total sum of card values for dealer
    int ptotal; //total sum of card values for player
    int dc; //difference from 21 of card total for dealer
    int pc; //difference from 21 of card total for player
    int chk; //either 1 or -1
    int dcp = 0; //number of dealer cards played
    int pcp = 0; //number of player cards played
    int aceChoice; //Holds the player's choice for the value of an ace

    //gets the first dealer card and chooses the value of aces to be 11
    getCard(dealer[0]);
    dcp++; //increment cards in play
    if(dealer[0].cardValue == 1)
        dealer[0].cardValue = 11;

    //gets the second dealer card and checks for/replaces repeat cards
```

```
getCard(dealer[1]);
dcp++; //increment cards in play
chk = check(dealer, player, dealer[1], dcp, pcp);
if(chk == -1)
{
    while(chk == -1)
    {
        getCard(dealer[1]);
        chk = check(dealer, player, dealer[1], dcp, pcp);
    }
}
//if an ace is drawn, 11 is chosen for its value if the dealer's total doesn't exceed 21
if(dealer[0].cardValue == 1)
    if(dealer[0].cardValue + 11 <= 21)
        dealer[0].cardValue = 11;


//gets player's first card and checks/replaces repeats
getCard(player[0]);
pcp++; //increment cards in play
chk = check(dealer, player, player[0], dcp+1, pcp);
while(chk == -1)
{
    getCard(player[0]);
```

```
    chk = check(dealer, player, player[0], dcp+1, pcp);
}

//gets players second card and checks/replaces repeats
getCard(player[1]);
pcp++; //increment cards in play
chk = check(dealer, player, player[1], dcp+1, pcp);
while(chk == -1)
{
    getCard(player[1]);
    chk = check(dealer, player, player[1], dcp+1, pcp);
}

//outputs the first dealer card and the two player cards
cout << "The dealers shown card is: ";
cout << dealer[0].cardName << ' ' << dealer[0].suit << endl << endl;
cout << "Your cards are: " << endl;
cout << player[0].cardName << ' ' << player[0].suit << endl;
cout << player[1].cardName << ' ' << player[1].suit << endl << endl;

//if the player draws a(n) ace(s), gets the players choice of value of either 1 or 11
if(player[0].cardValue ==1)
{
```

```
        aceChoice = aceGet();
            if(aceChoice == 11)
                player[0].cardValue = 11;
    }
    if(player[1].cardValue ==1)
    {
        aceChoice = aceGet();
            if(aceChoice == 11)
                player[1].cardValue = 11;
    }

    //gets the player's choice to draw another card or to keep current cards
    cout << "Would you like to hit or stay?" << endl;
    cin >> choice;

    //if choice is not "hit" or "stay" outputs an error message and prompts another input
for choice
    while(choice != "hit" && choice != "stay")
    {
        cin.clear();
        cout << "Please enter either hit or stay" << endl;
        cin >> choice;
    }
```

```
    if(choice == "stay") //executes if choice is "stay"
    {
        cout << endl;
        ptotal = player[0].cardValue + player[1].cardValue; //finds the player's current total
        cout << "The Dealer's second card is: " << endl;
        cout << dealer[1].cardName << ' ' << dealer[1].suit << endl << endl; //outputs
dealer's second card
        int dh = 2; //array position for dealer's next drawn cards
        dtotal = dealer[0].cardValue + dealer[1].cardValue; //finds dealer's current total

        while(dtotal <= 16) //only executes if the dealer total is less than or equal to 16 and
loops until the dealer total is above 16
        {
            cout << "The dealer draws a card" << endl;
            getCard(dealer[dh]); //gets dealers next card
            dcp++; //iterates the number of dealer cards in play
            //checks for/replaces repeat cards
            chk = check(dealer, player, dealer[dh], dcp, pcp+1);
            while(chk == -1)
            {
                getCard(dealer[dh]);
```

```
            chk = check(dealer, player, dealer[dh], dcp, pcp+1);
        }

        //if an ace is drawn, 11 is chosen for its value if the dealer's total doesn't exceed
21
        if(dealer[dh].cardValue == 1)
            if(dtotal + 11 <= 21)
                dtotal += 10;
        cout << "The next card is " << endl;
        cout << dealer[dh].cardName << ' ' << dealer[dh].suit << endl << endl; //outputs
the dealer's next card
        dtotal += dealer[dh].cardValue; //adds the card value to their total
        dh++; //iterates dealer structure array position
    }
  }



  int ph = 2; //array position for player's next card drawn


  if(choice == "hit") //only executes if choice is "hit"
  {
      ptotal = player[0].cardValue + player[1].cardValue; //finds the player's current total
```

```
        while(choice == "hit") //only executes if choice is "hit" and loops until choice is
"stay"
        {
            getCard(player[ph]); //gets player's next card
            pcp++; //iterates the number of player cards in play

            //checks for /replaces repeat cards
            chk = check(dealer, player, player[ph], dcp+1, pcp);
            while(chk == -1)
            {
                getCard(player[ph]);
                chk = check(dealer, player, player[ph], dcp+1, pcp);
            }
            cout << "You drew " << player[ph].cardName << ' ' << player[ph].suit << endl;
//outputs the player's drawn card

            //if the player draws an ace, gets the players choice of value of either 1 or 11
            if(player[ph].cardValue == 1)
            {
                aceChoice = aceGet();
                if(aceChoice == 11) //if the player chooses 11, changes the card value to 11
                    player[ph].cardValue = 11;
            }
```

```
        ptotal += player[ph].cardValue; //adds the card value to the player total

        if(ptotal >= 21) //makes choice "stay" if their total reaches or exceeds 21
            choice = "stay";

        if(ptotal < 21) // only executes if the player total is less than 21
        {
            //gets the player's choice to draw another card or to keep current cards
            cout << "Would you like to hit or stay?" << endl;
            cin >> choice;

            //if choice is not "hit" or "stay" outputs an error message and prompts another
input for choice
            while(choice != "hit" && choice != "stay")
            {
                cin.clear();
                cout << "Please enter either hit or stay" << endl;
                cin >> choice;
            }
        }
        ph++; //iterates the player structure array position
    }
```

```
    cout << endl;
    cout << "The Dealer's second card is: " << endl;
    cout << dealer[1].cardName << ' ' << dealer[1].suit << endl << endl; //outputs
dealer's second card
    int dh = 2; //array position for dealer's next drawn cards
    dtotal = dealer[0].cardValue + dealer[1].cardValue; //finds dealer's current total
    while(dtotal <= 16) //only executes if the dealer total is less than or equal to 16 and
loops until the dealer total is above 16
    {
        cout << "The dealer draws a card" << endl;
        getCard(dealer[dh]); //gets dealers next card
        dcp++; //iterates the number of dealer cards in play
        //checks for/replaces repeat cards
        chk = check(dealer, player, dealer[dh], dcp, pcp+1);
        while(chk == -1)
        {
            getCard(dealer[dh]);
            chk = check(dealer, player, dealer[dh], dcp, pcp+1);
        }

        //if an ace is drawn, 11 is chosen for its value if the dealer's total doesn't exceed
21
        if(dealer[dh].cardValue == 1)
```

```
            if(dtotal + 11 <= 21)
                dtotal += 10;
        cout << "The next card is " << endl;
        cout << dealer[dh].cardName << ' ' << dealer[dh].suit << endl << endl; //outputs
the dealer's next card
        dtotal += dealer[dh].cardValue; //adds the card value to their total
        dh++; //iterates dealer structure array position
    }
  }



  dc = 21-dtotal; //finds the difference of dealer's total and 21
  pc = 21-ptotal; //finds the difference of player's total and 21

  //Finds and outputs results of game as well as writes to gameResults structure
  *gameResults = results(dc, pc, dtotal, ptotal);

  char resultMessage[7]; //character array to hold the result message
  char gameName[9] = {'b', 'l', 'a', 'c', 'k', 'j', 'a', 'c', 'k'}; //character array to hold game
name
  char *readArray1; //pointer to hold copy of result message
  readArray1 = new char;
  char *readArray2; //pointer to hold copy of game name
```

```
readArray2 = new char;


if(gameResults->wr.winc == win) //stores "winner" in resultMessage
{
    resultMessage[0] = 'w';
    resultMessage[1] = 'i';
    resultMessage[2] = 'n';
    resultMessage[3] = 'n';
    resultMessage[4] = 'e';
    resultMessage[5] = 'r';
}
if(gameResults->wr.winc == lose) //stores "loser" in resultMessage
{
   resultMessage[0] = 'l';
   resultMessage[1] = 'o';
   resultMessage[2] = 's';
   resultMessage[3] = 'e';
   resultMessage[4] = 'r';
}
if(gameResults->wr.winc == draw) //stores "draw" in result Message
{
   resultMessage[0] = 'd';
   resultMessage[1] = 'r';
```

```
        resultMessage[2] = 'a';
        resultMessage[3] = 'w';
    }

    readArray1 = binaryFileManip("array.bin", resultMessage, sizeof(resultMessage));
//writes reaultMessage to array.bin file and copies it to readArray1
    readArray2 = binaryFileManip("array2.bin", gameName, sizeof(gameName)); //writes
gameName to array2.bin file and copies it to readArray2
    binaryRecord(player[0]); //writes player[0] to record.bin
    //outputs readArray2
    for(int i = 0; i < 9; i++)
        cout << readArray2[i];

    delete gameResults; //frees gameResult's memory
    delete[] readArray1; //frees readArray1's memory
    delete[] readArray2; //frees readArray2's memory

    //exits program
    return 0;
}



void getCard(struct card &c) //draws a card
```

```
{
    int n = rand() % 13 + 1; //gets a random number between 1 and 13
    int s = rand() % 4 + 1; //gets a random number between 1 and 4
    switch(n) //assigns a card value and name depending on the random number drawn
    {
        case 1: c.cardName = "Ace"; //assigns ace for 1
                c.cardValue = 1; //sets card value to 1
        break;
        case 2: c.cardName = "Two"; //assigns two for 2
                c.cardValue = 2; //sets card value to 2
        break;
        case 3: c.cardName = "Three"; //assigns three for 3
                c.cardValue = 3; //sets card value to 3
        break;
        case 4: c.cardName = "Four"; //assigns four for 4
                c.cardValue = 4; //sets card value to 4
        break;
        case 5: c.cardName = "Five"; //assigns five for 5
                c.cardValue = 5; //sets card value to 5
        break;
        case 6: c.cardName = "Six"; //assigns six for 6
                c.cardValue = 6; //sets card value to 6
        break;
```

```
    case 7: c.cardName = "Seven"; //assigns seven for 7
            c.cardValue = 7; //sets card value to 7
    break;
    case 8: c.cardName = "Eight"; //assigns eight for 8
            c.cardValue = 8; //sets card value to 8
    break;
    case 9: c.cardName = "Nine"; //assigns nine for 9
            c.cardValue = 9; //sets card value to 9
    break;
    case 10: c.cardName = "Ten"; //assigns ten for 10
             c.cardValue = 10; //sets card value to 10
    break;
    case 11: c.cardName = "Jack"; //assigns jack for 11
             c.cardValue = 10; //sets card value to 10
    break;
    case 12: c.cardName = "Queen"; //assigns queen for 12
             c.cardValue = 10; //sets card value to 10
    break;
    case 13: c.cardName = "King"; //assigns king for 13
            c.cardValue = 10; //sets card value to 10
    break;
}
```

```
    switch(s)
    {
        case 1: c.suit = "Spades"; //assigns suit to spades for 1
        break;
        case 2: c.suit = "Clubs"; //assigns suit to clubs for 2
        break;
        case 3: c.suit = "Diamonds"; //assigns suit to diamonds for 3
        break;
        case 4: c.suit = "Hearts"; //assigns suit to hearts for 4
        break;
    }
}

int check(struct card d[], struct card p[], struct card c, int dcp, int pcp) //checks if card
has been already been played
{
    for(int i = 0; i < dcp-1; i++) //loop iterates through every dealer card played before and
compares it to the card being checked
        if(c.cardName == d[i].cardName && c.suit == d[i].suit)
            return -1; //returns -1 if the card matches a previously played card
    if(pcp > 0) //only executes if there has been at least on player card played so far
        for(int i = 0; i < pcp-1; i++) //loop iterates through every player card played before
and compares it to the card being checked
```

```
        if(c.cardName == p[i].cardName && c.suit == p[i].suit)
            return -1; //returns -1 if the card matches a previously played card
    return 1; //return 1 if no repeat cards were found
}

int aceGet() //gets the value of ace that the player decides
{
    int ac; // holds ace choice
    cout << "Would you like the Ace to be worth 1 or 11?" << endl;
            cin >> ac; //gets the ace choice

            //loops if ace choice is not 1 or 11, outputting an error message and
prompting another input for ace choice
            while(ac != 1 && ac != 11 || cin.fail())
            {
                cin.clear();
                cin.ignore();
                cout << "Please enter a valid choice" << endl;
                cin >> ac;
            }
    return ac; //return the choice for ace
}
```

```
struct winTracker results(int dc, int pc, int dtotal, int ptotal) //outputs game results and
records the result
{
    struct winTracker s; //defines the struct to return
    char result[] = "You lose"; //creates a cstring for the lose message

    cout << "Dealer's " << dtotal << " vs " << "Player's " << ptotal << endl; //outputs the
dealer and player totals

    if(dc == 0 && pc == 0 && dc != pc) //outputs no winner if both total differences are
zero
    {
        cout << "No winner" << endl;
        s.wr.winc = draw; //sets winc to draw
    }
    if(dc == pc) //outputs no winner if both total differences are the same
    {
        cout << "No winner" << endl;
        s.wr.winc = draw; //sets winc to draw
    }
    if(dc < 0 && pc < 0) //outputs no winner if both total differences are negative
    {
        cout << "No winner" << endl;
```

```
        s.wr.winc = draw; //sets winc to draw
    }

    if(dc < 0 && pc >= 0) //outputs dealer busts you win if dealer total difference is
negative and the player's is not
    {
        cout << "Dealer busts" << endl;
        cout << "You win" << endl;
        s.wr.winc = win; //sets winc to win
    }
    if(dc >= 0 && pc < 0) //outputs you bust you lose if player total difference is negative
and dealer's is not
    {
        cout << "You bust" << endl;
        cout << "You lose" << endl;
        s.wr.winc = lose; //sets winc to lose
    }
    if(dc < pc && dc >= 0) //outputs lose message cstring if dealer total differnce is
smaller than player's and not negative
    {
        cout << result << endl;
        s.wr.winc = lose; //sets winc to lose
    }
```

```
    if(pc < dc && pc >= 0) //outputs you win if player total difference is smaller than
dealer's and is not negative
    {
        cout << "You win" << endl;
        s.wr.winc = win; //sets winc to win
    }

    return s; //return the structure holding the result
}

char *binaryFileManip(string fileName, char *inputArray, int arrSize) //reads and writes
to binary files
{
    char *arr; //defines char pointer to hold a char array
    arr = new char[10];
    fstream file(fileName, ios::in | ios::out | ios::binary); //defines a file stream for input
and output to a binary file
    file.seekp(0L, ios::beg); //goes to start of file to write
    file.write(inputArray, arrSize); //writes the passed array to the file
    file.seekg(0L, ios::beg); //goes to the start of the file to read from
    file.read(arr, sizeof(arr)+1); //reads from the file and stores in the char pointer

    file.close(); //close the file
```

```
    return arr; //return the memory address of the char pointer
}

void binaryRecord(struct card s) //writes a card structure to a binary file
{
    fstream file("record.bin", ios::out | ios::binary); //defines a file stream for input to
record.bin
    file.seekp(0L, ios::beg); //goes to the start of the file
    file.write(reinterpret_cast<char*>(&s), sizeof(s)); //writes the passed structure to
record.bin

    file.close(); //close the file
}
```