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
}
```