Project 1

Tittle

Three Card Game

Course

CSC-17A

Section

48096

Due Date

October 28, 2016

Author

Sili Guo

Table of Contents

I. Introduction

Three Card Game is a traditional poker gabbling game that is popular in China. Its rules is similar to many of western card games, but it is more flexible about the number of people playing this game. And the game between two people and several people can experience different feeling, and use different strategies during the game.

I like poker games because I enjoy the feeling of guessing others' cards by the action they do during the game; the process of holding small cards to try to bluff makes me feel excited. However, after playing many western poker games, such as Texas hold'em, and Black jack, I miss the day I played Three Card Game with my Chinese friends, for it makes me feel more about relax, rather than nervous.

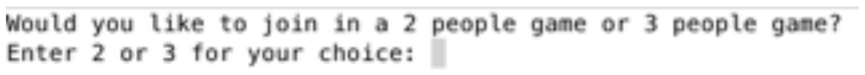
The most interesting part for this game is there's always a way to win. The rules of the smallest cards set of 2, 3, and 5 can beat the largest cards set of full house! You will never know if you can win others or not before open both cards sets.

Note: In this game, 0 represent to 10.

II. Game Rules

A. Decide players number

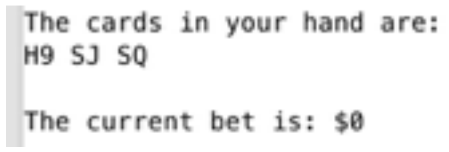
The Three Card Game Program only limit 2~3 players to play this game. At the beginning, the program will ask player if he wants to join a 2 people game, or 3 people game



```
Would you like to join in a 2 people game or 3 people game?
Enter 2 or 3 for your choice:
```

B. Discard

After choose the number of players, the game started. Player can see the cards in his hand, and the bet of the table currently.

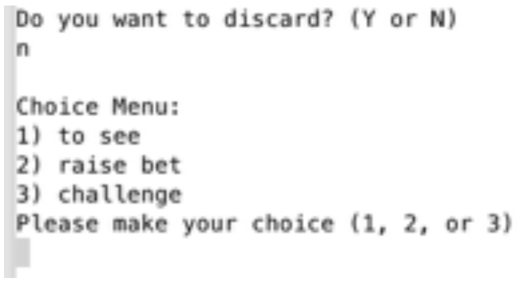


```
The cards in your hand are:
H9 SJ SQ

The current bet is: $0
```

Game starts by randomly pick the dealer. At the beginning of each round, the system will ask player if he would like to discard his cards. If the player was luckily picked as the dealer, then he will has one chance to leave the table without paying anything; otherwise, player can not take the bet on the table back.

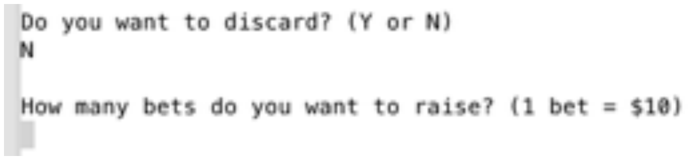
C. Player’s Action

If the player decide to play the game with his cards, the system will automatically display the choice menu of player’s action: to see, raise bet, or challenge previous player.



```
Do you want to discard? (Y or N)
n

Choice Menu:
1) to see
2) raise bet
3) challenge
Please make your choice (1, 2, or 3)
```

Note: During the first round of the game, if the dealer did not discard, he can only choose to raise bet.

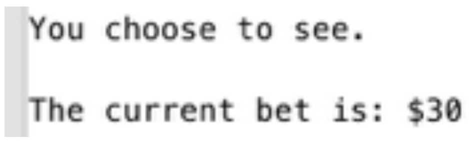


```
Do you want to discard? (Y or N)
N

How many bets do you want to raise? (1 bet = $10)
```

D. To See

To see is a simply strategy in card games to see how other people react, which may implicit if he has a good cards set or not.

By choosing to see, the player’s bet will be equal to the current bet on the table, and the current bet will not change.

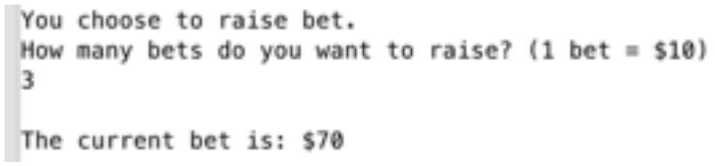


```
You choose to see.

The current bet is: $30
```

E. Raise Bet

If a player choose to raise bet, that means he wants to increase the amount of the bet on the table. This can be regarding as a signal of good cards set; but also can be a good strategy of bluff during the game.

To raise bet, there will always be a minimum and maximum amount for the bet to raise. For this program, player can raise at least 1 bet($10), but at most 10 bets($100) for each round; and the table can hold totally 120 bets ($1200);



```
You choose to raise bet.
How many bets do you want to raise? (1 bet = $10)
3

The current bet is: $70
```

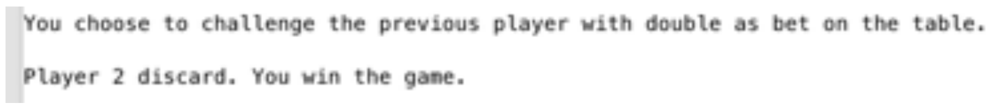
F. Challenge Previous Player

The most interesting part of this game is to enjoy the process of challenging others and waiting to be challenged.

Challenge in Three Card Game only allow two players to compare(the compare rules will be discussed under this section); and there will be a judge to look at both players' card and report the winner, which means a player will not know others card even his challenge success.

When a player decide to challenge the previous player, he must has at least twice as much money as the bet of the table currently, because challenging others will cost twice as much as bet for their private bet. The winner takes all money, and loser exit game.

When a player is challenged, he can choose to discard and give the bet on the table to the challenger, or he can choose to bet the same amount of challenger to compare their cards.



```
You choose to challenge the previous player with double as bet on the table.
Player 2 discard. You win the game.
```

G. Compare Rules

There are seven ranks of cards set in Three Card Game:

- Full House (three same cards)

- Straight Flush
- Flush
- Straight
- One pair
- Single
- Special (2, 3, 5 in dif suit)

The rules are simple: full house > straight flush > flush > straight > one pair > single > special > full house.

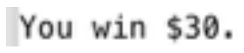
Remind that special is the smallest card set in the game, but it can beat full house.

If two players have cards in same rank, then compare from the largest card to smallest one.

Since Three Card Game do not count suit, if two players have exactly the same cards, the challenger is the loser.

H. Result

If the player lose during the challenge, the game is over; otherwise, output how much the player won or lost.



```
You win $30.
```

III. Summary

Total Line of Code	**908**
Comment Line	144
Variable	20
Function	8

Note: Some of the comment line are used for testing.

The total function based on the structure Player. I stored all the information about one player in the pointer of Player type. To make sure the data is correct, I write it into a binary file, and read it when necessary. The program is constructed with decision makings. I used lots of loop and switch case to help program decide the branches that game will go. The AI makes decisions based on the random number creation. To pass variables between main and function, I use pointer point to function, pass by address, and pass pointer as variables.

VI. Challenge During Program

A. Logic Errors

Since this is a large program, I struggled a lot on which process goes first and which goes second. Even after I finished the program, there are some logic mistakes that I spend a lot of time to correct.

However, by this problem, I learned how to write the plan before converting it to codes. It can be easier to see when you write on a piece of paper rather than write code directly on IDE.

B. How to Control AI Decision

To make a game that let player play with AI is one of the hardest thing I met in this program. A card game like this depends on a lot for how player react during game. It is the problem that how can I make the AI do what people will do.

This ask me to predict what people will do during the game, and make AI to do the same thing. However, how to achieve this goal is a hard problem. After discussing with my friends in same major, we come up a way to use random number to control the percent of what an AI may do during the game.

C. Conversion Between integer and character

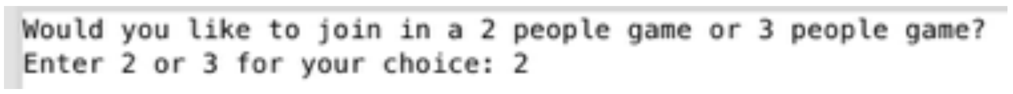
This is also a problem that I spend lots time on. The way I create a poker (52 cards without jokers) is first to create a number array of 52 elements filled with numbers 1~13. Then I convert the number array to the array of char to represent the number (eg. A, J, Q, K). However, the array of char gives me a lot of problems when comparing two card sets.

To solve this problem, I tried several ways, but finally decide to convert the char back to int after 3 cards are dealt to player, and stored in the players information as well. To make the compare easier, I also sort the cards that player hold from smallest to largest.

V. Description

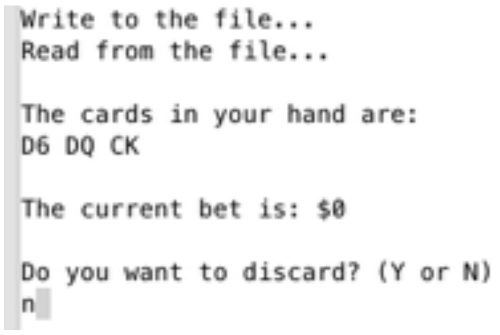
A. Sample Input/Output

First decide the players number



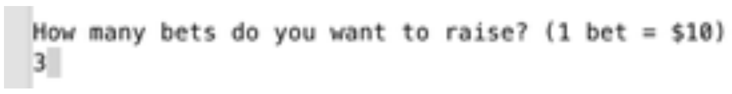
```
Would you like to join in a 2 people game or 3 people game?
Enter 2 or 3 for your choice: 2
```

The player is luckily chose as dealer. Ask if player if he would like to discard



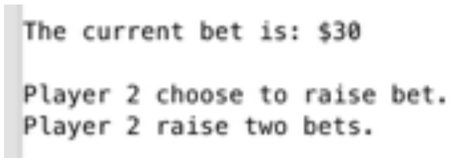
```
Write to the file...
Read from the file...

The cards in your hand are:
D6 DQ CK

The current bet is: $0

Do you want to discard? (Y or N)
n
```

For first round, only ask to raise bet



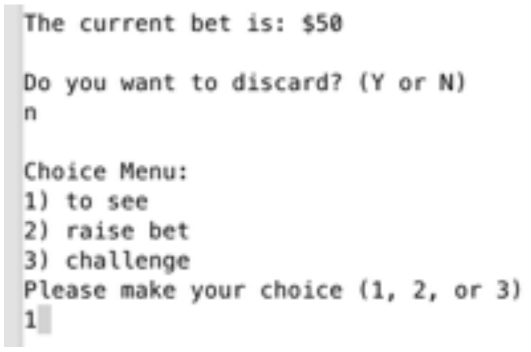
```
How many bets do you want to raise? (1 bet = $10)
3
```

Then round for AI



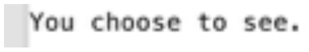
```
The current bet is: $30

Player 2 choose to raise bet.
Player 2 raise two bets.
```

Player’s 2nd round, ask if discard
If no, output choice menu



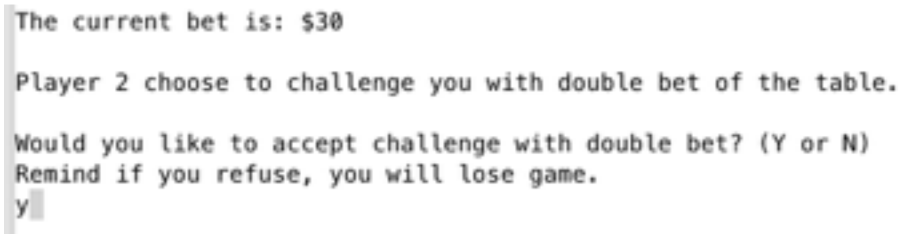
```
The current bet is: $50

Do you want to discard? (Y or N)
n

Choice Menu:
1) to see
2) raise bet
3) challenge
Please make your choice (1, 2, or 3)
1
```

Choose 1 means to see



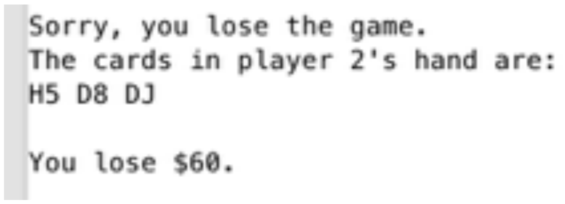
```
You choose to see.
```

AI choose to challenge player



```
The current bet is: $30

Player 2 choose to challenge you with double bet of the table.

Would you like to accept challenge with double bet? (Y or N)
Remind if you refuse, you will lose game.
y
```

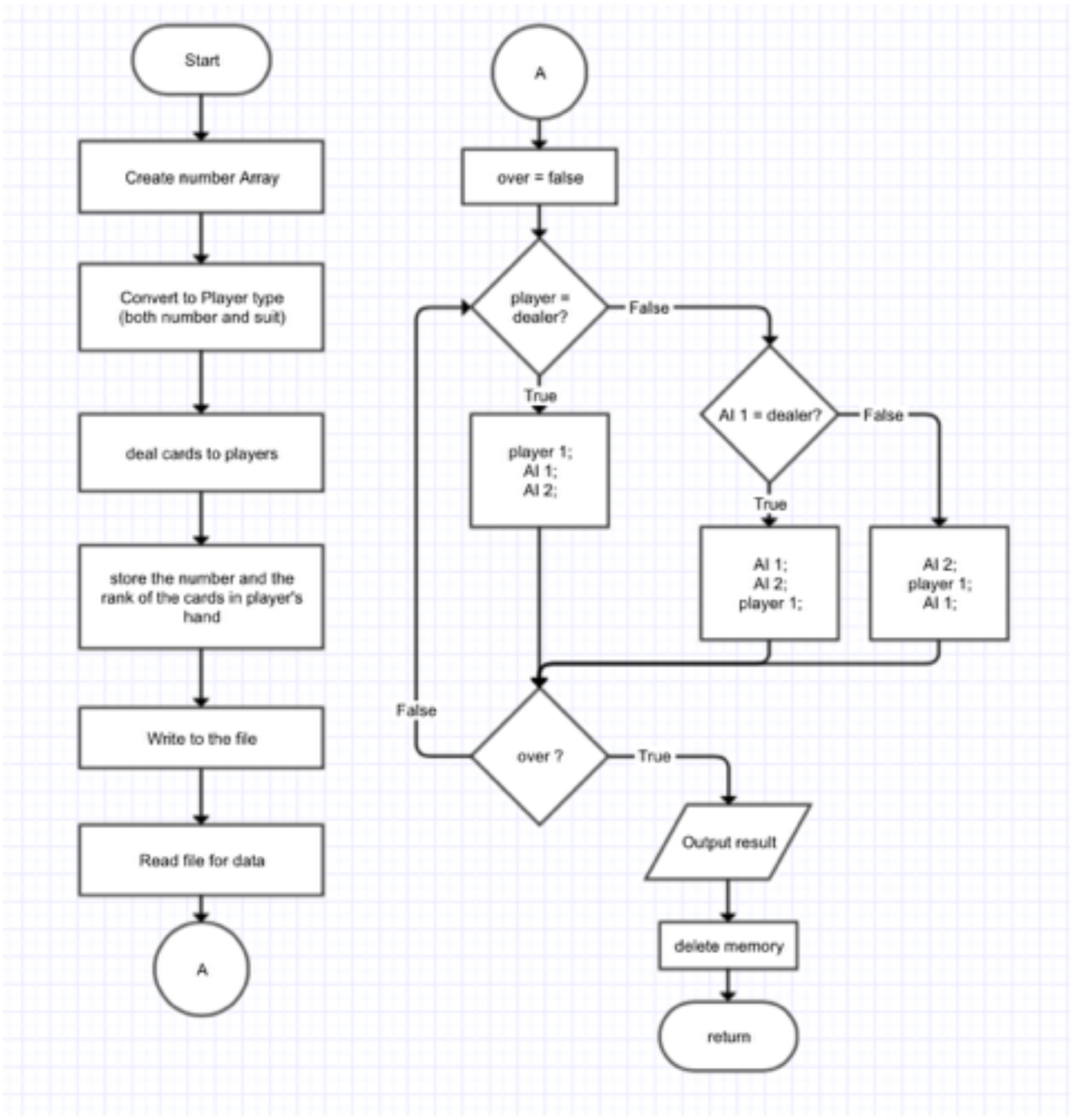
Accept challenge, and see the result



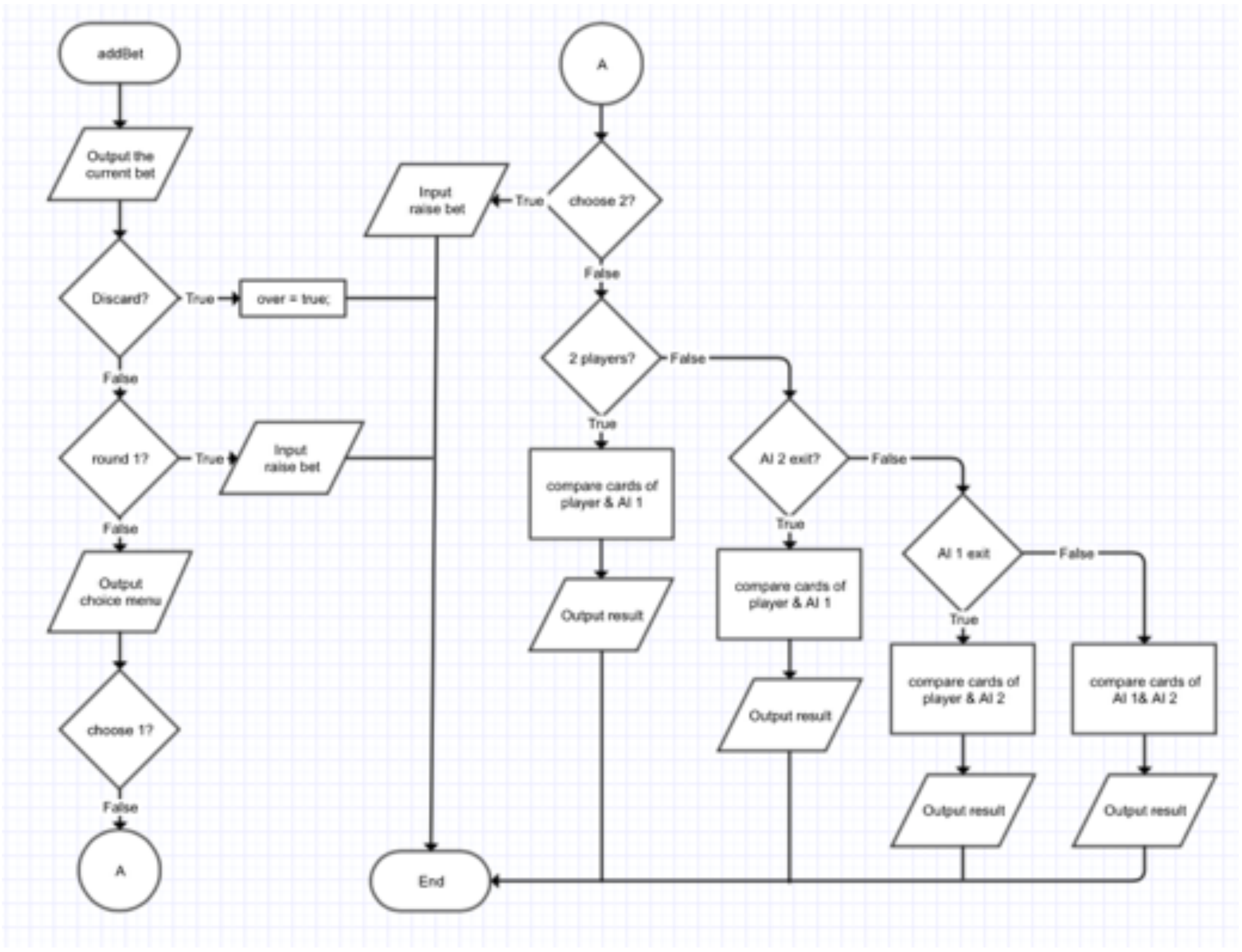
```
Sorry, you lose the game.
The cards in player 2's hand are:
H5 D8 DJ

You lose $60.
```

B. Flowcharts

Main:



addBet Function:



C. Pseudocode

```
• create a number array contains the number of 2~14
• convert the number array to Card array, containing suit and number
• promote the players number from player
• deal 3 cards to each player
• convert the 3 cards to number, and calculate the rank
• store all data about player into Player *players
• write the data into the file
• read the data from file
• randomly choose a player as dealer
• If player:
    • ask if discard
    • during first round, promote for the raised bet
    • not first round, output choice menu
    • if 1: to see —— do nothing, calculate money
    • if 2 raise bet —— promote raised bet, calculate money
    • if 3 challenge ——
        • if 2 players —— compare player & AI 1
```

        - if 3 players ——
            - if AI 2 exit —— compare player & AI 1
            - else —— compare player & AI 2
- if AI:
    - AI I never discard
    - if 3 players —— AI 2 10% discard
    - if AI 2 not exit
    - if first round —— AI randomly raise bet
    - if not first —— randomly choose from 1~3 (1:2:2)
    - if choose 1 —— calculate money
    - if choose 2 —— randomly raise bet
    - if choose 3 ——
        - if AI 1:
            - if 2 players —— compare player & AI 1
            - if 3 players ——
                - if AI 2 exit, game over
                - else continue
        - If AI 2:

    - if AI 1 exit —— compare player & AI 2
    - else —— compare AI 1 & AI 2
- calculate the money of player
- output result (win or lose challenge, win or lose money)

D. Variables

Type	Variable Name	Description	Declare Location(line)
const int	Size	size of the number array	33
int	number[SIZE]	the number array of cards	34
	swap	represent the suit	34
	r	hold a random number	108
	choice	the number of players	126
	sortNum	temp hold number for sort	164
	start	random choose dealer	260
	max_one	max bet in one round	268
	max	max bet in the game	269
	round	the round	270
	bet	the bet on the table	271
	money	the money player holds after game	311
bool	mak	sign to stop sort loop	165
	over	game over	263
	two_exit	AI 1 exit game	264
	thr_exit	AI 2 exit game	265
Card	num[SIZE]	array of cards(suit & number)	35
	temp	temp hold data of 1 card	107
Player *	players	array holds all players' data	138

E. Concepts

Concept	Type	Code	Location(line)
Binary file	fstream	ios::out | ios::binary ios::in | ios::binary	894 904
Structure array	Card	Card num[SIZE];	35
Dynamic structure array	Player *	Player *players = new Player[choice];	138
Structure in a structure	Player	Card * hold;	Player.h

Return structure from function	Card *
	Player *
		Card *dealCard(Card *, int);
		Player *playerInitial(Card *, int);
			21
			22

Structure as function argument	Player *	void addBet(Player *, bool&, bool&, bool&, int&, int&, int);	23
Pointer to structure	Card	Card * hold;	Player.h

Bubble sort	int	sort = players[i].hold[j];
		players[i].hold[j] = players[i].hold[j + 1];
		players[i].hold[j + 1] = sort;
			162-183

Switch	switch	switch(start)	275
Loop	for	for (int i = 0; i < SIZE; i++)	79
	do-while	do {} while(mak)	165
Type casting	reinterpret_cast <type>	reinterpret_cast<char *> (p)	896

Random number	rand()	if (rand() % 5 < 3)	621

VI. Code

```
/*
 * File:   main.cpp
 * Author: Sili Guo
 * Created on October 27, 2016
 * Purpose: Project 1 --- Three Card Game
 */


#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <ctime>
using namespace std;


//User's library
#include "Card.h"
#include "Player.h"


//Function Prototypes
//void wholeCards(Card []);
```

```
Player *playerInitial(Card *, int);

Card *dealCard(Card *, int);

void addBet(Player *, bool&, bool&, bool&, int&, int&, int);

void addBetAI(Player *, int, bool&, bool&, bool&, int&, int&, int);

bool challenge(Player);

bool compare(Player, Player);

void wtFile(Player *, int);

void rdFile(Player *, int);


int main(int argc, char** argv) {


    //Declare variables

    const int SIZE = 52; //The size of a set of poker cards (without joker)

    int number[SIZE], swap = 0;

    Card num[SIZE];


    //Create a seed for random numbers

    srand(static_cast<unsigned int> (time(0)));


    //Create an array that contains 52 cards (without joker)

    //Get the number of the cards, and fill the suit of each card in array

    for (int i = 0, j = 1; i < SIZE; i++, j++) {
```

```
    //fill the array with number of 1~13
    number[i] = j + 1;
    //restart count for j, when higher than 13
    if (j == 13) {
        j = 0;
    }
    //Decide the suit of the number
    swap = i / 13;
    switch (swap) {
        case(0):
            num[i].suit = 'D'; //The first row are Diamonds
            break;
        case(1):
            num[i].suit = 'C'; //The second row are Clubs
            break;
        case(2):
            num[i].suit = 'H'; //The third row are Hearts
            break;
        case(3):
            num[i].suit = 'S'; //The fourth row are Spades
    }
}//End of for loop
```

```
    //    //
******************************************************************************
*************************************
    //    //For test: display the number array
    //    cout << "For test: The number array is: " << endl;
    //    for (int i = 0; i < SIZE; i++) {
    //        cout << number[i] << " ";
    //        if ((i + 1) % 13 == 0) {
    //            cout << endl;
    //        }
    //    }
    //    cout << endl;
    //    //
******************************************************************************
*************************************

    //Fill the card numbers of array and Decide the special condition of numbers
    for (int i = 0; i < SIZE; i++) {
        num[i].number = number[i] + '0'; //switch int into char
        switch (i % 13) {
            case(12):
```

```
            num[i].number = 'A'; //1 show as A
            break;
        case(8):
            num[i].number = '0'; //10 show as 0
            break;
        case(9):
            num[i].number = 'J'; //11 show as J
            break;
        case(10):
            num[i].number = 'Q'; //12 show as Q
            break;
        case(11):
            num[i].number = 'K'; //13 show as K
      }
  }

  //   //
*****************************************************************************
************************************
  //   //For test: show the initial poker card array
  //   cout << "For test: The initial card array: " << endl;
  //   wholeCards(num);
  //    cout << endl;
  //    //
*******************************************************************************
*************************************

  //Shuffle cards
  Card temp; //Temporarily hold the data of a card
  int r; //Hold a random number
  for (int i = 0; i < SIZE; i++) {
      //Randomly pick an card
      r = rand() % SIZE;
      //Replace cards between each two
      temp = num[r];
      num[r] = num[i];
      num[i] = temp;
  }

  //    //
*******************************************************************************
*************************************
  //    //For test: show the poker card array after shuffled
  //    cout << "For test: The card array after shuffled: " << endl;
  //    wholeCards(num);

  //    cout << endl;

  //    //
*******************************************************************************
*************************************


  //Promote the number of players from user

  int choice;

  do {

      cout << "Would you like to join in a 2 people game or 3 people game?" << endl;

      cout << "Enter 2 or 3 for your choice: ";

      cin >> choice;

      if (choice != 2 && choice != 3) {

          cout << "Invalid Input! Please enter a number between 2 and 3." << endl;

      }

  } while (choice != 2 && choice != 3);

  cout << endl;


  //Allocate a dynamic array of players based on the user's choice

  Player *players = new Player[choice];


  //Game begin: deal three cards to each player
```

```
players = playerInitial(num, choice);


//Convert char back to int
for (int i = 0; i < choice; i++) {
    for (int j = 0; j < 3; j++) {
        if (players[i].hold[j].number == 'A') {
            players[i].holdNum[j] = 14;
        } else if (players[i].hold[j].number == '0') {
            players[i].holdNum[j] = 10;
        } else if (players[i].hold[j].number == 'J') {
            players[i].holdNum[j] = 11;
        } else if (players[i].hold[j].number == 'Q') {
            players[i].holdNum[j] = 12;
        } else if (players[i].hold[j].number == 'K') {
            players[i].holdNum[j] = 13;
        } else {
            players[i].holdNum[j] = players[i].hold[j].number - 48;
        }
    }
}


//Sort the card set in players' hand, and the number array
```

```
Card sort;

int sortNum;

bool mak;

do {

    mak = false;

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

        for (int j = 0; j < 2; j++) {

            if (players[i].holdNum[j] > players[i].holdNum[j + 1]) {

                //Sort card in char

                sort = players[i].hold[j];

                players[i].hold[j] = players[i].hold[j + 1];

                players[i].hold[j + 1] = sort;

                //Sort card number

                sortNum = players[i].holdNum[j];

                players[i].holdNum[j] = players[i].holdNum[j + 1];

                players[i].holdNum[j + 1] = sortNum;

                mak = true;

            }//End of if statement

        }//End of for j loop

    }//End of for i loop

} while (mak);
```

```
    //Decide the rank of each player's card
    for (int i = 0; i < choice; i++) {
        //Check rank of 0
        if (players[i].holdNum[0] == 2 && players[i].holdNum[1] == 3 && players[i].holdNum[2]
== 5) {
            players[i].rank = 0;
        }
        //Check rank of 1
        if (players[i].holdNum[0] != players[i].holdNum[1] && players[i].holdNum[1] !=
players[i].holdNum[2]) {
            players[i].rank = 1;
        }
        //Check rank of 2
        if ((players[i].holdNum[0] == players[i].holdNum[1] && players[i].holdNum[0] !=
players[i].holdNum[2]) ||
                (players[i].holdNum[1] == players[i].holdNum[2] && players[i].holdNum[0] !=
players[i].holdNum[1])) {
            players[i].rank = 2;
        }
        //Check rank of 3, 4 & 5
        if (players[i].hold[0].suit == players[i].hold[1].suit && players[i].hold[1].suit ==
players[i].hold[2].suit) {
```

```
            if (players[i].holdNum[1] + 1 == players[i].holdNum[1] && players[i].holdNum[1] + 1
== players[i].holdNum[2]) {
                players[i].rank = 5;
            } else {
                players[i].rank = 4;
            }
        } else {
            if (players[i].holdNum[0] + 1 == players[i].holdNum[1] && players[i].holdNum[1] + 1
== players[i].holdNum[2]) {
                players[i].rank = 3;
            }
        }
        //Check rank of 6
        if (players[i].holdNum[0] == players[i].holdNum[1] && players[i].holdNum[1] ==
players[i].holdNum[2]) {
            players[i].rank = 6;
        }
    }//End block of for loop

    //    //
**************************************************************************
***********************************
```

```
//    //For test: show the number array after sort
//    cout << "For test: The cards set is: " << endl;
//    for (int i = 0; i < choice; i++) {
//        for (int j = 0; j < 3; j++) {
//            cout << players[i].holdNum[j] << " ";
//        }
//        cout << endl;
//    }
//    cout << endl;
//    //For test: show all three set of cards after sort
//    cout << "For test: The cards set is: " << endl;
//    for (int i = 0; i < choice; i++) {
//        for (int j = 0; j < 3; j++) {
//            cout << players[i].hold[j].suit << players[i].hold[j].number << " ";
//        }
//        cout << endl;
//    }
//    cout << endl;
//    //For test: show the rank after sort
//    cout << "For test: The rank set is: " << endl;
//    for (int i = 0; i < choice; i++) {
//
//        cout << players[i].rank << " ";
//    }
//    cout << endl << endl;
//    //
*****************************************************************************
************************************

    //Write players info in binary file
    wtFile(players, choice);

    //Read the info from the file
    rdFile(players, choice);

    //Display user's card
    cout << "The cards in your hand are: " << endl;
    for (int i = 0; i < 3; i++) {
        cout << players[0].hold[i].suit << players[0].hold[i].number << " ";
    }
    cout << endl << endl;

    //Randomly chose dealer to start the game
    int start = rand() % choice;
```

```
//Set sign for different meaning

bool over = false; //game over

bool two_exit = false; //player 2 lose game and exit

bool thr_exit = false; //player 3 lose game and exit


//Set money limit for one round, and for the whole game

int max_one = 100; //Each person maximumly bet 300 in one round

int max = 1200; //If total bet reach $3000, force all players open their cards

int round = 0; // sign for the first round

int bet = 0;


//Loop the game till

do {

   switch (start) {

      case(0):

         if (!over) {//If game not over

            addBet(players, over, two_exit, thr_exit, round, bet, choice);

         }

         if (over) {

            break;

         } else {
```

```
        round++;
    }


case(1):
    if (!over && !two_exit) {
        addBetAI(players, choice, over, two_exit, thr_exit, round, bet, 1);
    }
    if (over) {
        break;
    } else {
        round++;
    }
case(2):
    if (choice == 3) {
        if (!over && !thr_exit) {
            addBetAI(players, choice, over, two_exit, thr_exit, round, bet, 2);
        }
        if (over) {
            break;
        } else {
            round++;
        }
```

```
            }
        }//End block of switch case
        start = 0;
    } while (!over);

    //Output the status of player's money
    int money = players[0].money + players[0].bet;
    cout << endl;
    if (money > 1000) {
        cout << "You win $" << money - 1000 << "." << endl;
    } else if (money < 1000) {
        cout << "You lose $" << 1000 - money << "." << endl;
    } else {
        cout << "You neither win or lose." << endl;
    }

    //Delete memory
    delete [] players;
    players = 0;

    return 0;
}
```

```
//Function implementation

////For test: Function show the whole poker card array

//

//void wholeCards(Card *n) {

//    for (int i = 0; i < 52; i++) {

//        cout << n[i].suit << n[i].number << " ";

//        if ((i + 1) % 13 == 0) {

//            cout << endl;

//        }

//    }

//}


//Initialize the information of each player


Player *playerInitial(Card *n, int c) {

    //Allocate dynamic array

    Player * pl = new Player[c];


    //fill the array

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

        pl[i].hold = dealCard(n, i);
        pl[i].money = 1000; //Each player has $1000 at beginning
        pl[i].bet = 0; //How much each player bet on table
    }
    //Return array
    return pl;
}

//Deal three cards to each player

Card *dealCard(Card *n, int c) {
    //Allocate dynamic array
    Card *ptr = new Card[3];
    for (int i = 0; i < 3; i++) {
        ptr[i].number = n[3 * c + i].number;
        ptr[i].suit = n[3 * c + i].suit;
    }
    //Return array
    return ptr;
}

//Ask user if he wants to add bet
```

```
void addBet(Player *p, bool& o, bool& two, bool& thr, int& r, int& b, int n) {
    //Declare variables
    string ans;
    int choice, add;

    //Display current bet
    cout << "The current bet is: $" << b << endl << endl;

    //Ask user if he wants to discard
    do {
        cout << "Do you want to discard? (Y or N)" << endl;
        cin >> ans;

        //Check validation
        if (ans != "Y" && ans != "N" && ans != "y" && ans != "n") {
            cout << "Invalid Input! Please enter Y or N." << endl;
        }
    } while (ans != "Y" && ans != "N" && ans != "y" && ans != "n");
    cout << endl;

    if (ans == "Y" || ans == "y") {//If player discard, game over
        o = true;
        p[0].bet = 0;
        cout << "You discard your cards. Game over." << endl;
    } else if (ans == "N" || ans == "n") {//If player choose to continue play, ask him to add bet
        if (r == 0) {//For first round
            do {
                cout << "How many bets do you want to raise? (1 bet = $10)" << endl;
                cin >> add;
                //Check validation
                if (add < 1) cout << "Invalid Input! You have to at least raise one bet." << endl;
                if (add > 10) cout << "Invalid Input! You can at most raise 10 bets in one round." <<
endl;
            } while (add < 1 || add > 10);
            cout << endl;
            //Minus bet from your money, add it to bet on the table
            b += add * 10;
            p[0].money = 1000 - b;
            p[0].bet = b;
        } else {//For other round
            cout << "Choice Menu:" << endl;
            cout << "1) to see" << endl;
            cout << "2) raise bet" << endl;
            cout << "3) challenge" << endl;
            cout << "Please make your choice (1, 2, or 3)" << endl;

            cin >> choice;

            cout << endl;


            //Use switch case for choice

            switch (choice) {

                case(1):

                   cout << "You choose to see." << endl << endl;

                   p[0].money -= b;

                   p[0].bet = b;

                   break;

                case(2):

                   do {

                      cout << "You choose to raise bet." << endl;

                      cout << "How many bets do you want to raise? (1 bet = $10)" << endl;

                      cin >> add;

                      //Check validation

                      if (add < 1) cout << "Invalid Input! You have to at least raise one bet." << endl;

                      if (add > 10) cout << "Invalid Input! You can at most raise 10 bets in one round."
<< endl;

                   } while (add < 1 || add > 10);

                   cout << endl;
```

```
            //Minus bet from your money, add it to bet on the table
            b += add * 10;
            p[0].money -= b;
            p[0].bet = b;
            break;
        case(3):
            if (p[0].money < 2 * b) {
                cout << "You do not have enough money to challenge others." << endl;
                cout << endl;
            } else {
                cout << "You choose to challenge the previous player with double as bet on the
table." << endl;
                cout << endl;
                //AI decision
                if (n == 2) {
                    //If AI accept
                    if (challenge(p[1])) {
                        if (compare(p[0], p[1])) {
                            o = true;
                            p[0].money += 2 * b;
                            p[1].money -= b;
                            p[1].bet = 0;
            cout << "Congratulation! You win the game!" << endl;
        } else {
            o = true;
            p[0].money -= b;
            p[0].bet = 0;
            p[1].money += 2 * b;
            cout << "Sorry, you lose the game." << endl;
        }
        cout << "The cards in player 2's hand are: " << endl;
        for (int i = 0; i < 3; i++) {
            cout << p[1].hold[i].suit << p[1].hold[i].number << " ";
        }
        cout << endl;
    } else {
        o = true;
        p[0].money += b;
        p[1].bet = 0;
        cout << "Player 2 discard. You win the game." << endl;
    }
} else if (n == 3) {
    if (!thr) {
        if (challenge(p[2])) {
```

```
if (compare(p[0], p[2])) {
    if (!two) {
        thr = true;
        cout << "Congratulation! You win the challenge!" << endl;
        cout << "Player 3 exit the game." << endl;
        cout << endl;
    } else {
        o = true;
        cout << "Congratulation! You win the game." << endl;
        cout << "The cards in player 3's hand are: " << endl;
        for (int i = 0; i < 3; i++) {
            cout << p[2].hold[i].suit << p[2].hold[i].number << " ";
        }
        cout << endl;
    }
    p[0].money += 2 * b;
    p[2].money -= b;
    p[2].bet = 0;
} else {
    o = true;
    p[0].money -= b;
    p[0].bet = 0;
```

```
            p[2].money += 2 * b;
            cout << "Sorry, you lose the game." << endl;
            cout << "The cards in player 3's hand are: " << endl;
            for (int i = 0; i < 3; i++) {
                cout << p[2].hold[i].suit << p[2].hold[i].number << " ";
            }
            cout << endl;
        }
    } else {
        if (!two) {
            thr = true;
            cout << "Player 3 discard, and exit the game." << endl;
            cout << endl;
        } else {
            o = true;
            cout << "Player 3 discard, and exit the game." << endl;
            cout << "Congratulation! You win the game." << endl;
        }
        p[0].money += b;
        p[2].bet = 0;
    }
} else {
        if (challenge(p[1])) {
            if (compare(p[0], p[1])) {
                o = true;
                p[0].money += 2 * b;
                p[1].money -= b;
                p[1].bet = 0;
                cout << "Congratulation! You win the game!" << endl;
            } else {
                o = true;
                p[0].money -= b;
                p[0].bet = 0;
                p[1].money += 2 * b;
                cout << "Sorry, you lose the game." << endl;
            }
            cout << "The cards in player 2's hand are: " << endl;
            for (int i = 0; i < 3; i++) {
                cout << p[1].hold[i].suit << p[1].hold[i].number << " ";
            }
            cout << endl;
        }
    }
}
```

```
                }


            }//End block of switch case

        }//Not the first round

    }//user choose to continue play

}


//Function for AI player 2


void addBetAI(Player *p, int n, bool& o, bool & two, bool& thr, int& r, int& b, int i) {

    //Declare variables

    int choice;

    string ans;


    //Display current bet

    cout << "The current bet is: $" << b << endl << endl;


    //AI decision: discard

    if (n == 3) {

        if (p[2].rank == 1) {

            if (rand() % 10 < 1) {

                thr = true;
            cout << "Player 3 discard, and exit game." << endl;
            cout << endl;
        }
    }
}
//AI decision: add Bet
if (!thr) {
    if (r == 0) {
        if (p[i].rank == 0 || p[i].rank == 1 || p[i].rank == 2) {
            if (rand() % 5 < 3) {
                cout << "Player " << i + 1 << " raise one bet." << endl;
                b += 10;
            } else {
                cout << "Player " << i + 1 << " raise five bets." << endl;
                b += 50;
            }
        } else if (p[i].rank == 3 || p[i].rank == 4) {
            if (rand() % 2 < 1) {
                cout << "Player " << i + 1 << " raise one bet." << endl;
                b += 10;
            } else {
                cout << "Player " << i + 1 << " raise three bets." << endl;
            b += 30;
        }
    } else {
        if (rand() % 5 < 2) {
            cout << "Player " << i + 1 << " raise three bet." << endl;
            b += 30;
        } else {
            cout << "Player " << i + 1 << " raise five bets." << endl;
            b += 50;
        }
    }
    cout << endl;
    p[i].money = 1000 - b;
    p[i].bet = b;
} else {
    if (rand() % 5 < 1) {
        choice = 1;
    } else {
        if (rand() % 2 < 1) {
            choice = 2;
        } else {
            choice = 3;
    }
}
switch (choice) {
    case(1):
        cout << "Player " << i + 1 << " choose to see." << endl << endl;
        p[i].money = 1000 - b;
        p[i].bet = b;
        break;
    case(2):
        cout << "Player " << i + 1 << " choose to raise bet." << endl;
        if (p[i].rank == 0 || p[i].rank == 1 || p[i].rank == 2) {
            if (rand() % 5 < 3) {
                cout << "Player " << i + 1 << " raise one bet." << endl;
                b += 10;
            } else {
                if (rand() % 4 < 3) {
                    cout << "Player " << i + 1 << " raise two bets." << endl;
                    b += 20;
                } else {
                    cout << "Player " << i + 1 << " raise five bets." << endl;
                    b += 50;
                }
            }
        } else if (p[i].rank == 3 || p[i].rank == 4) {
            if (rand() % 2 < 1) {
                cout << "Player " << i + 1 << " raise two bet." << endl;
                b += 20;
            } else {
                cout << "Player " << i + 1 << " raise three bets." << endl;
                b += 30;
            }
        } else {
            if (rand() % 5 < 2) {
                cout << "Player " << i + 1 << " raise three bet." << endl;
                b += 30;
            } else {
                if (rand() % 6 < 5) {
                    cout << "Player " << i + 1 << " raise five bets." << endl;
                    b += 50;
                } else {
                    cout << "Player " << i + 1 << "raise ten bets." << endl;
                    b += 100;
                }
            }
        }
        cout << endl;
        p[i].money -= b;
        p[i].bet = b;
        break;
      case(3):
        if (i == 1) {
          if (n == 2) thr = true;
          cout << "Player 2 choose to challenge you with double bet of the table." << endl
<< endl;
          do {
            cout << "Would you like to accept challenge with double bet? (Y or N)" <<
endl;
            cout << "Remind if you refuse, you will lose game." << endl;
            cin >> ans;
            //Check validation
            if (ans != "Y" && ans != "N" && ans != "y" && ans != "n") {
              cout << "Invalid Input! Please enter Y or N." << endl;
            }
          } while (ans != "Y" && ans != "N" && ans != "y" && ans != "n");
          cout << endl << endl;
```

```
if (ans == "N" || ans == "n") {
    o = true;
    p[i].money += b;
    p[i - 1].bet = 0;
    cout << "You choose to discard." << endl;
    cout << "Sorry, you lose the game." << endl;
} else {
    if (compare(p[i], p[i - 1])) {
        o = true;
        p[i].money += 2 * b;
        p[i - 1].money -= b;
        p[i - 1].bet = 0;
        cout << "Sorry, you lose the game." << endl;
        cout << "The cards in player 2's hand are: " << endl;
        for (int i = 0; i < 3; i++) {
            cout << p[1].hold[i].suit << p[1].hold[i].number << " ";
        }
        cout << endl;
    } else {
        if (!thr) {
            two = true;
            cout << "You win the challenge!" << endl;
```

```
                cout << "Player 2 exit game." << endl;
                cout << endl;
            } else {
                o = true;
                cout << "Player 2 lose the challenge." << endl;
                cout << "Congratulation! You win the game!" << endl;
            }
            p[i].money -= b;
            p[i].bet = 0;
            p[i - 1].money += 2 * b;
        }
    }
}
if (i == 2) {
    if (!two) {
        cout << "Player 3 choose to challenge player 2 with double bet as table." <<
endl;
        cout << endl;
        if (!challenge(p[1])) {
            two = true;
            p[i].money += b;
            p[i - 1].bet = 0;
    cout << "Player 2 refuse challenge from player 3." << endl;
    cout << "Player 2 exit game." << endl;
    cout << endl;
} else {
    if (compare(p[i], p[i - 1])) {
        two = true;
        p[i].money += 2 * b;
        p[i - 1].money -= b;
        p[i - 1].bet = 0;
        cout << "Player 2 lose the challenge with player 3." << endl;
        cout << "Player 2 exit the game." << endl;
        cout << endl;
    } else {
        thr = true;
        p[i].money -= b;
        p[i].bet = 0;
        p[i - 1].money += 2 * b;
        cout << "Player 3 lose the challenge with player 2." << endl;
        cout << "Player 3 exit the game." << endl;
        cout << endl;
    }
}
```

```
        } else {
            cout << "Player 3 choose to challenge you." << endl;
            do {
                cout << "Would you like to accept challenge with double bet? (Y or N)" <<
endl;
                cout << "Remind if you refuse, you will lose game." << endl;
                cin >> ans;
                //Check validation
                if (ans != "Y" && ans != "N" && ans != "y" && ans != "n") {
                    cout << "Invalid Input! Please enter Y or N." << endl;
                }
            } while (ans != "Y" && ans != "N" && ans != "y" && ans != "n");
            if (ans == "N" || ans == "n") {
                o = true;
                p[i].money += b;
                p[i - 2].bet = 0;
                cout << "You choose to discard." << endl;
                cout << "Sorry, you lose the game." << endl;
            } else {
                if (compare(p[i], p[i - 2])) {
                    o = true;
                    p[i].money += 2 * b;
                    p[i - 2].money -= b;
                    p[i - 2].bet = 0;
                    cout << "Sorry, you lose the game." << endl;
                    cout << "The cards in player 3's hand are: " << endl;
                    for (int i = 0; i < 3; i++) {
                        cout << p[2].hold[i].suit << p[2].hold[i].number << " ";
                    }
                    cout << endl;
                } else {
                    o = true;
                    p[i].money -= b;
                    p[i].bet = 0;
                    p[i - 2].money += 2 * b;
                    cout << "Player 3 lose the challenge." << endl;
                    cout << "Congratulation! You win the game!" << endl;
                }
            }
        }
    }
  }
 }
}
```

```
}


//Function of AI decision for challenge


bool challenge(Player p) {
    //Declare variables
    bool accept = false;


    //AI make decision
    //Rank 0: 1% accept
    if (p.rank == 0) {
        if (rand() % 100 < 1) accept = true;
    } else if (p.rank == 1) {//Rank 1
        if (p.holdNum[2] < 8) {//2~7, not accept
            accept = false;
        } else if (p.holdNum[2] >= 8 && p.holdNum[2] < 11) {//8~10, 20% accept
            if (rand() % 5 < 1) accept = true;
        } else {//>10, 40% accept
            if (rand() % 5 < 2) accept = true;
        }
    } else if (p.rank == 2) {//Rank 2
        if (p.holdNum[1] < 7) {//A pair of 1~7, 50% accept
```

```
            if (rand() % 2 < 1) accept = true;
        } else {//Otherwise, 60% accept
            if (rand() % 5 < 3) accept = true;
        }
    } else if (p.rank == 3) {//Rank 3
        if (p.holdNum[2] < 9) {//Straight under 7, 8, 9, 80% accept
            if (rand() % 5 < 4) accept = true;
        } else {//Otherwise, 90% open
            if (rand() % 10 < 9) accept = true;
        }
    } else {//If rank higher than 3, 100% accept
        accept = true;
    }
    return accept;
}


//Compare cards of two players


bool compare(Player p, Player q) {
    //Declare variables
    bool win = false;
    if (p.rank == 0 && q.rank == 6) {
        win = true;
    } else if (p.rank == 6 && q.rank == 0) {
        win = false;
    } else {
        if (p.rank > q.rank) {
            win = true;
        } else if (p.rank < q.rank) {
            win = false;
        } else {//Rank equals
            //If same cards
            if (p.holdNum[0] == q.holdNum[0] && p.holdNum[1] == q.holdNum[1] &&
p.holdNum[2] == q.holdNum[2]) {
                win = false;
            } else {//Not same cards
                if (p.rank == 3 || p.rank == 5 || p.rank == 6) {//Rank 3, 5, 6
                    if (p.holdNum[2] > q.holdNum[2]) {
                        win = true;
                    } else {
                        win = false;
                    }
                } else if (p.rank == 1 || p.rank == 4) {//Rank 2, 4
                    if (p.holdNum[2] > q.holdNum[2]) {
                        win = true;
                    } else if (p.holdNum[2] < q.holdNum[2]) {
                        win = false;
                    } else {
                        if (p.holdNum[1] > q.holdNum[1]) {
                            win = true;
                        } else if (p.holdNum[1] < q.holdNum[1]) {
                            win = false;
                        } else {
                            if (p.holdNum[0] > q.holdNum[0]) {
                                win = true;
                            } else {
                                win = false;
                            }
                        }
                    }
                } else if (p.rank == 2) {
                    if (p.holdNum[1] > q.holdNum[1]) {
                        win = true;
                    } else if (p.holdNum[1] < q.holdNum[1]) {
                        win = false;
```

```
                } else {
                    if (p.holdNum[2] > q.holdNum[2]) {
                        win = true;
                    } else if (p.holdNum[2] < q.holdNum[2]) {
                        win = false;
                    } else {
                        if (p.holdNum[1] > q.holdNum[1]) {
                            win = true;
                        } else {
                            win = false;
                        }
                    }
                }
            }
        }//Not same cards
    }//Rank equals
  }
  return win;
}

//Write the info into the file
```

```
void wtFile(Player *p, int n) {
   fstream output;
   cout << "Write to the file..." << endl;
   output.open("players.txt", ios::out | ios::binary);
   if (!output.fail()) {
      output.write(reinterpret_cast<char *> (p), sizeof (Player) * n);
   }
   output.close();
}

void rdFile(Player *p, int n) {
   fstream input;
   cout << "Read from the file..." << endl << endl;
   input.open("players.txt", ios::in | ios::binary);
   if (!input.fail()) {
      input.read(reinterpret_cast<char *> (p), sizeof (Player) * n);
   }
   input.close();
}
```