Edwin Kam

Personal Project

C++ Blackjack simulator

(computer vs computer) //we just sit and wait for the result

```
19      void shuffle(int*, size_t); //shuffle card function
20      void create(int*, const int);//poker card generator
21      void print(int*, const int);//function that print the the poker deck in order
22      void printf(string*, const int);//string function that print the the poker de
23      int aceaction(int);//when player or dealer get ace
24      int splitaction(int);//when player can split
25      int paction(int);//player action
26      int intdis();//distributing card to player and dealer
27      void play(int);//play function
28      void dealeraction();//dealer action function
```

Objective: This program will stimulate the result for numerous games by using our adjustable strategy. By recording stats from computer vs computer while adjusting our bet, we can generate a must-win formula. This program will tell you, when you have more chance to win, and how you should adjust your bet!

The only reason I created this program was because I want to verify whether the must-win formula mentioned in Ben Mezrich's book *Bring Down the House* exists. *

4-8 Decks, Dealer Stands on Soft 17

Player	Dealer's card									
hard	2	3	4	5	6	7	8	9	10	A
4-8	H	H	H	H	H	H	H	H	H	H
9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	Rh	H
16	S	S	S	S	S	H	H	Rh	Rh	Rh
17+	S	S	S	S	S	S	S	S	S	S
soft	2	3	4	5	6	7	8	9	10	A
13	H	H	H	Dh	Dh	H	H	H	H	H
14	H	H	H	Dh	Dh	H	H	H	H	H
15	H	H	Dh	Dh	Dh	H	H	H	H	H
16	H	H	Dh	Dh	Dh	H	H	H	H	H
17	H	Dh	Dh	Dh	Dh	H	H	H	H	H
18	S	Ds	Ds	Ds	Ds	S	S	H	H	H
19+	S	S	S	S	S	S	S	S	S	S
splits	2	3	4	5	6	7	8	9	10	A
2,2	Ph	Ph	P	P	P	P	H	H	H	H
3,3	Ph	Ph	P	P	P	P	H	H	H	H
4,4	H	H	H	Ph	Ph	H	H	H	H	H
6,6	Ph	P	P	P	P	H	H	H	H	H
7,7	P	P	P	P	P	P	H	H	H	H
8,8	P	P	P	P	P	P	P	P	P	P
9,9	P	P	P	P	P	S	P	P	S	S
A,A	P	P	P	P	P	P	P	P	P	P

wizardofodds.com

H	Hit
S	Stand
Dh	Double if allowed, otherwise hit
Ds	Double if allowed, otherwise stand
P	Split
Ph	Split if double after split is allowed, otherwise hit
Rh	Surrender if allowed, otherwise hit

IMPORTANT!!!!

This is a strategy chart that we can found online. By following this chart, the players can maximize their win rate. However, **the win rate of the dealer would still be higher than the player** because that's how blackjack designed.

However, we can still make money from the dealer if we can determine whether we are in an advantage situation.

In other words, you raise the bet when you know you have higher chance to win; lower the bet when you are more likely to lose.

This program will tell you, when you have more chance to win, and how you should adjust your bet!!!

Final result image

```
gamecount: 52    This set Used card count: 69
================================================================================
Playerwin game: 26       Playerwin rate: 0.5
Dealerwin game: 23       Dealerwin rate: 0.442308
TIE/split game: 3       Tie rate: 0.0576923
Double game: 4   Double rate: 0.0769231
Split game : 2  Split rate: 0.0384615
Player win: $ 56        Dealer win: $ 41
Player net winning: $15

C:\Users\kamsh\Desktop\blackjackmine\Debug\blackjackmine.exe (process 5892) exited with code 0.
Press any key to close this window . . .
```

The final stats include:

- Numbers of game we played
- How many cards we used (important because this program is based on statistic)
- Win and tie rate of the player and the dealer.
- Split and double rate

Procedure overview:

#1 Write a simple 1v1 blackjack program, including card shuffling, card distributing, simple game rules.

```
95          player[0] = intdis();//disturbiting cards
96          player[1] = intdis();
97          blackjack(player, "Player");//check if player has blackjack
98          int temp = value(player[0]) + value(player[1]);//sum of player hands
99          coutcard(player, "Player");//cout player hand
100         dealer[0] = intdis();//disturbiting cards
101         dealer[1] = intdis();
102         cout << "Dealer has " << display(dealer[0]) << endl;//dealer exposed card
103         coutcard(dealer, "Dealer");
104         if (blackjack(player, "Player") && !blackjack(dealer, "Dealer"))
105         {
106             cout << "Dealer no blackjack\n";
107             bet=bet*1.5;
108             cout << "Player gets $" << bet << endl;
109         }
110         else if (blackjack(dealer, "Dealer") && !blackjack(player, "Player"))
111         {
112             cout << "SORRY!!\n";
113         }
114         else if (!blackjack(dealer, "Dealer"))//if dealer no blackjack
115         {
116             if (player[0] == player[1] && player[0] < 10)//pairs except 10
117             {
118                 cout << "!!!!!!!!!!!!!!!!!!";
119                 play(splitaction(player[0]));//go to splite with that card
120             }
121             else if (checkace(player)) //if player has an ace
122             {
123                 play(aceaction(temp - 1));
124             }
```

```
Game 1
$$bet is $1
Player card: 8, 2,      Player sum: 10
Dealer has 7
Dealer card: 7, 4,      Dealer sum: 11
```

The program will display the player's hand, the dealer's exposed card and the dealer's hidden card for research purposes.

#2 Program the strategy chart to teach the computer how to deal with each situation.

```
327     switch (hand)
328     {
329     case 2:
330     case 3:
331         if (value(dealer[0]) <= 4 || value(dealer[0]) >= 7)//1-4, 7-10
332         {
333             //cout << "*******player called hit" << endl;
334             return 1;
335         }
336         else
337         {
338             //cout << "*******player called double" << endl;
339             return 2;
```

```
519         case 2: //double
520             doublerate++;
521             bet = bet * 2;//double the bet
522             cout << "*******player called double2" << endl;
523             cout << "$$bet of $" << bet << endl;
524             player[2] = intdis();
525             acevalue(player);//determine ace value
526             coutcard(player, "Player");
527             pbust(player);//check player busts
528             cout << "-------Dealer turn\n";
529             dealeraction();//dealer action
530             break;
531         case 3://stand
```

```
Game 1
$$bet is $1
Player card: 8, 2,       Player sum: 10
Dealer has 7
Dealer card: 7, 4,       Dealer sum: 11
*******player called double2
$$bet of $2
Player card: 8, 2, 10,   Player sum: 20
-------Dealer turn
Dealer card: 7, 4, 6,    Dealer sum: 17
Dealer stand
Dealer card: 7, 4, 6,    Dealer sum: 17
========Player won!!!!!
Player win: $ 2 Dealer win: $ 0
Player net winning: $2
```

The program can decide what to do as a player for every game now.

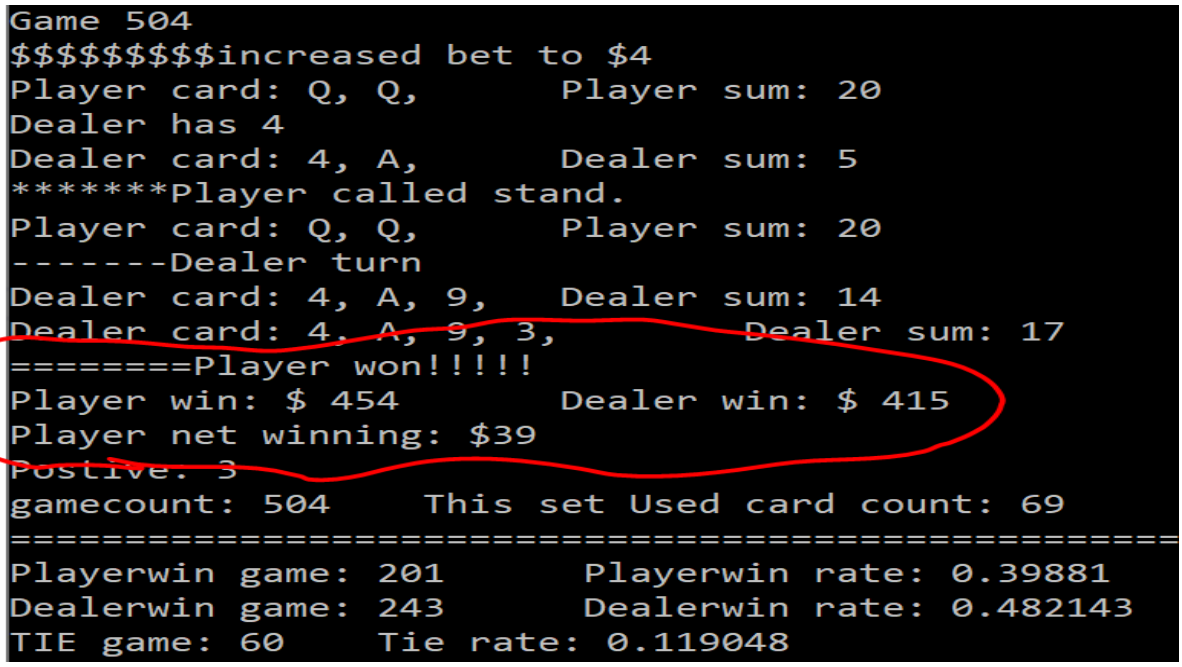
#3 We can now calculate the win rate of the player after about 500 games

```
Game 504
$$$$$$$$$increased bet to $4
Player card: Q, Q,       Player sum: 20
Dealer has 4
Dealer card: 4, A,       Dealer sum: 5
*******Player called stand.
Player card: Q, Q,       Player sum: 20
-------Dealer turn
Dealer card: 4, A, 9,    Dealer sum: 14
Dealer card: 4, A, 9, 3,         Dealer sum: 17
========Player won!!!!!
Player win: $ 454        Dealer win: $ 415
Player net winning: $39
Postive: 3
gamecount: 504     This set Used card count: 69
==========================================
Playerwin game: 201       Playerwin rate: 0.39881
Dealerwin game: 243       Dealerwin rate: 0.482143
TIE game: 60     Tie rate: 0.119048
```

The win rate of the player in 504 games is 40%.

PS. The maximum player win rate is about 40-43%. The strategy chart that we applied to this program already maximized the win rate. We will not be able to increase our win rate anymore.

#4 Even the player has a lower win rate, you can see the player is still winning.



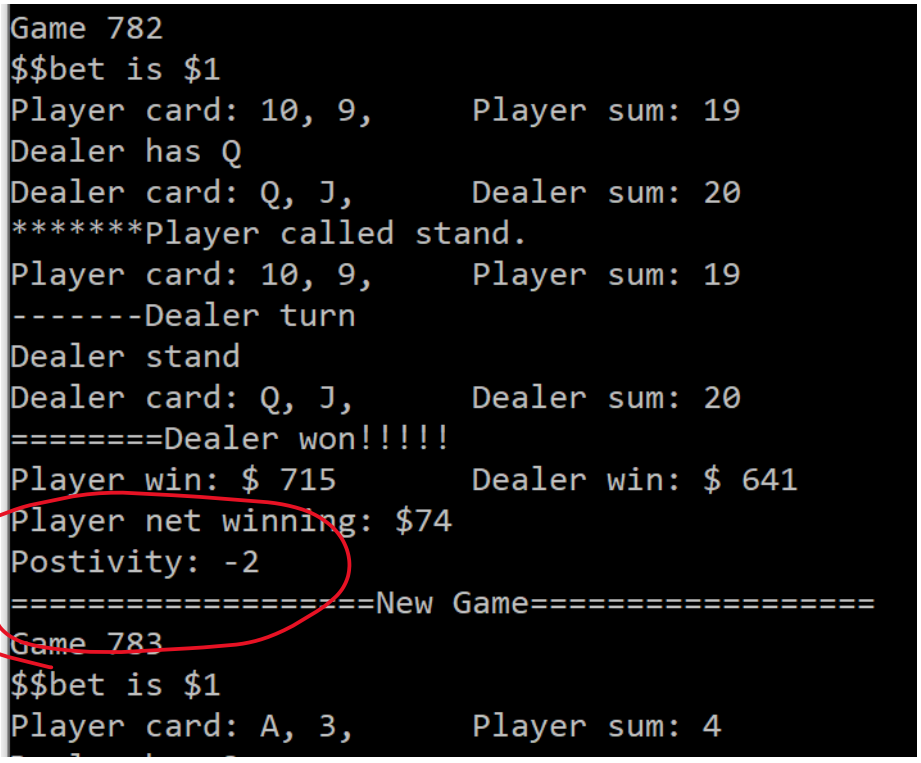
It is because we can change the bet when the situation is advantageous to us. For example, in Game 504, the system changed the bet to $4 automatically.

#5 What defines advantage situation. **Important!**

The more small value cards distributed, the more advantage we have. All cards that were distributed on the table must be shown to the players, so we can count them. We only need to identify whether smaller value cards distributed more than high value cards. In this program, the system will count for us.

```
int intdis()
{
    if (poker[cardcount] <= 6 && poker[cardcount] >=2)//2-6 positive
    {
        positive++;
    }
    else if (poker[cardcount] >9|| poker[cardcount] ==1)//10 and A negetive
    {
        positive--;
    }
    return poker[cardcount++];//distribute the next card
```

Positivity = # of distributed small value card (2-6) (minus-) # of distributed high value card (10, J, Q, K).

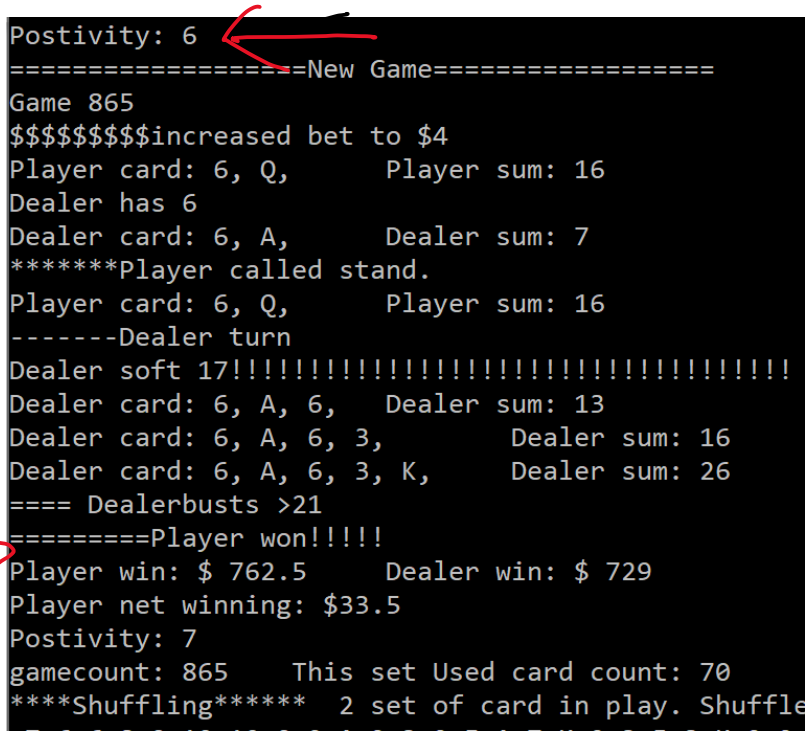


Positivity will be shown after each game.

Positive=advantage

negative=disadvantage

In game 782, the positivity is negative 2 (disadvantage) now, the program will only bet $1 for the next game.

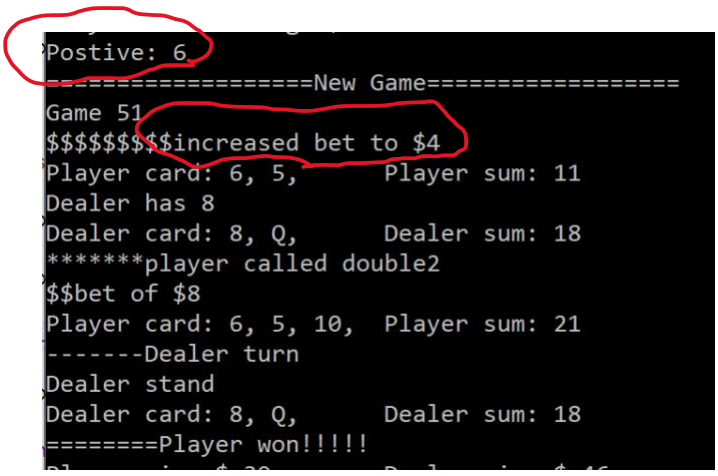


In game 865, the positivity was positive 6 by the end of last game. Theoretically, the player has a very high chance to win.

#6. How do we adjust the bet?

The positivity only indicates whether we have higher chance to win; it does not guarantee a win. Therefore, we need to develop our own bet strategy for how to adjust our bet. Changing the strategy is very easy; we can do it within 30 seconds. For example, we can double up the bet when positive = 3 and quadruple the bet when positive>3.

```
if (positive> 3)
{
    bet = 4;
    cout << "$$$$$$$$$increased bet to $" << bet << endl;
}
else if (positive ==3)
{
    bet = 2;
    cout << "$$$$$$$$$increased bet to $" << bet << endl;
}
else
{
    bet = 1;
    cout << "$$bet is $" << bet << endl;
}
```



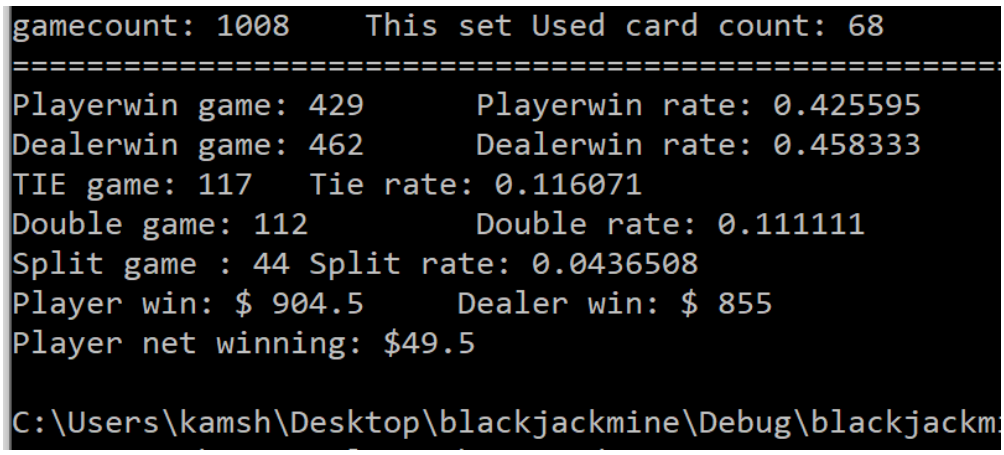
The system will automatically change the bet according to our bet strategy now.

Final step. Examine your strategy

The stats are more accurate if more games were played. This is the whole point of my simulator; The program can simulate a hundred thousand games in 25 minutes. You can verify whether your strategy makes a profit using this simulator.

```
const int numset = 2; //how many set of card will be use
double percent = 0.65;//how many cards used before before shuffling
int gamenum = 1008; //how many games you want
```

You can customize the card distribution because every casino has its own rule.



Conclusion: Using the example strategy in step six, the player earns $49.5 after 1008 games. Therefore, a must-win strategy does exist. According to Ben Mezrich's book *Bring Down the House*, there were 6 MIT students who generated a must-win formula using this method (counting positivity) and won a billion dollars. Therefore, if you think the profit is too small, you can always go back to step 6 and adjust your strategy.

The reason I created this program only because I want to verify whether the must-win formula mentioned in Ben Mezrich's book *Bring Down the House* exists. *

I hope this program demonstrate my creativity and programming skills. I will keep develop more creative ideas, especially ideas that can benefit the community such as idea related to health and science. If you have any questions regarding this simulator, please do not hesitate to contact me.