Project Report On Blackjack Minimum Risk Algorithm

Submitted by
S.Akshay Raj(180001058)
Prashant Kumar Rajak(180001037)
Computer Science and Engineering
2nd year

Under the Guidance of
Dr. Kapil Ahuja

Department of Computer Science and Engineering
Indian Institute of Technology Indore
Spring 2020

1) <u>INTRODUCTION</u> :

BLACKJACK is a card game the object of which is to be dealt with cards having a higher count than those of the dealer up to but not exceeding 21. It is probably the only casino game, in which you have a chance to beat the house with logic and strategy. To understand the algorithms and strategies on blackjack first we need to know the rules of it.

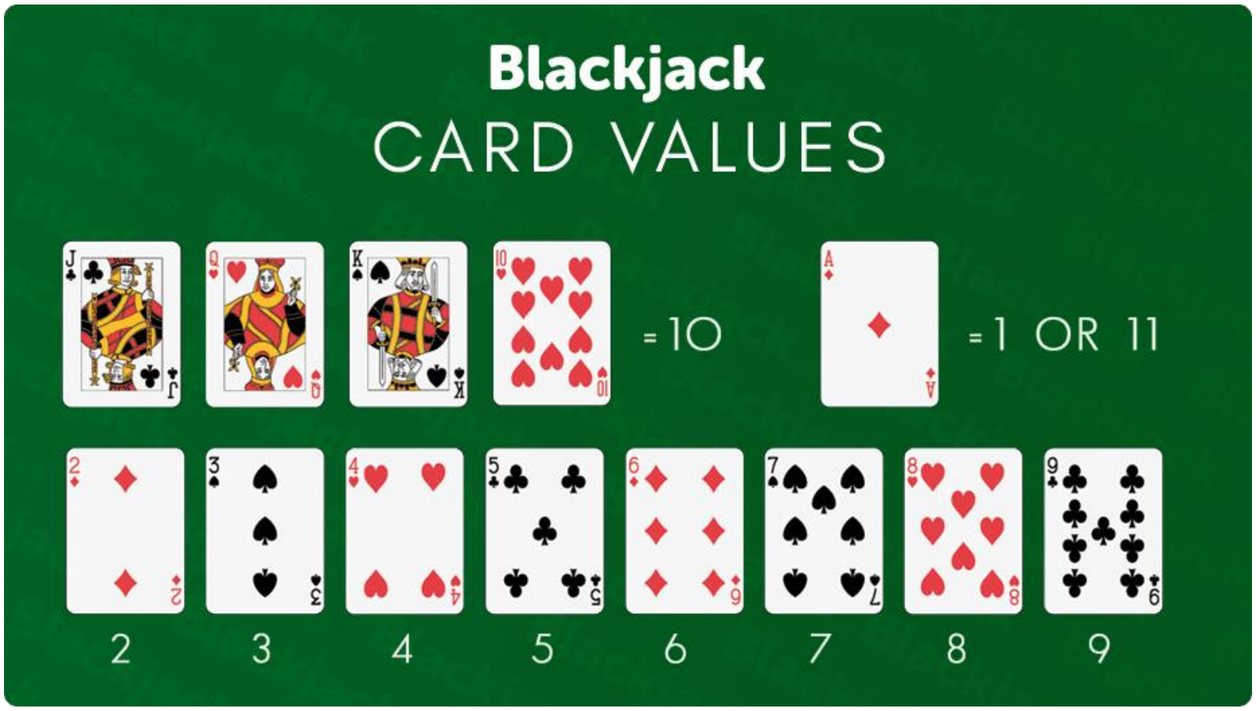
RULES OF BLACKJACK:

The dealer and each player receive two cards. The dealer turns the first of his cards face up and the other remains face down. Players' cards are typically dealt face up, but this is not necessary.

The object of the game is to come as close to 21 as possible without going over, which is termed “busting” Each card is counted as its face value, with face cards counting *10*, and aces being counted as 1 or 11, at the player's discretion. The dealer must follow fixed rules, which prescribe generally that the dealer must continue to take cards until reaching a total of 17 or higher. Dealer aces are counted generally as 11 unless it would cause the dealer to bust. The player, unlike the dealer, has the discretion to hit or stand, and in certain cases has additional options of “splitting” or “doubling down.”

Let’s say for example the player is dealt with two cards that are 10 of spades and 8 of hearts. And the dealer got 8 of diamonds and the other card is hidden for now. The player has got options like to hit, to stand, double down in which case he will be given only one card while doubling the bet. The optimum decision would be to stand as there is a high chance of getting busted as the total is 18. And now the dealer reveals his hidden card and has to hit until he reaches 17 or more so on and so forth. There is also a special case, when the player receives two same cards, with additional bet, same as previous, he can split them and play them separately. Double down can be used after splitting, though in some of the casinos, there is a limit to how many times a player can split or double down.

2) STRATEGIES :



After knowing the rules, let’s move to the strategy and gameplay part. High-speed computers were used to simulate each distinct situation in the game. The computer calculated the correct move for each element, and each of these correct moves was pieced back together to form an overall strategy. The strategies were defined based on either starting from a fresh deck or decks of cards - a so-called “basic strategy” - or reflecting the distribution of types of cards, such as low or high, or 10s or 5s, that had already been played in prior hands - so-called “counting strategies.”

And hence we come to the so-called basic strategy introduced by Thorp, who also wrote books for this to crack blackjack along with his colleagues. Even though this actually doesn’t help you win but it gives a slight percentage of winning money. After all, the basic strategy is designed so as to maximize profits over consecutive hands and shorten the losses incurred during this.

The strategy comprises essentially the “sum of the parts” decisions for each separate situation that can be considered. Thos slightly varies with the number of cards and type of shuffling. The basic strategy is designed so as to maximize profits over consecutive hands and shorten the losses incurred during the same.

The basic strategy has been evolved, modified, and named as evolved basic strategy, keeping in mind the changes incorporated in the game.

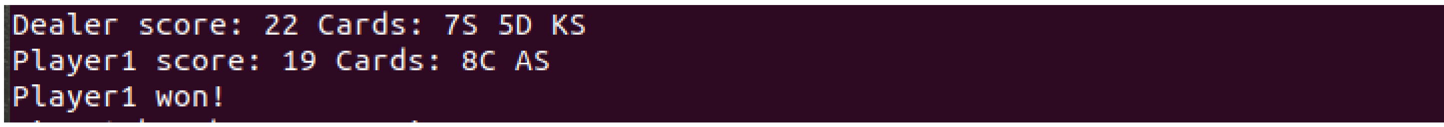
3) Algorithm Design : (modified Thorp’s Strategy)

Using Thorsp’s Modified Strategy -
The standard blackjack strategies rely on a separate analysis of the correct

play in each of a set of various possible situations. Each of these situations is viewed independently. When the correct plays are combined, by altering betting and the decisions are taken based on the composition of cards that have been played, the player can have a decided advantage over the house. The result is either the dealer is busted or has lesser hand values.

Gaming console-

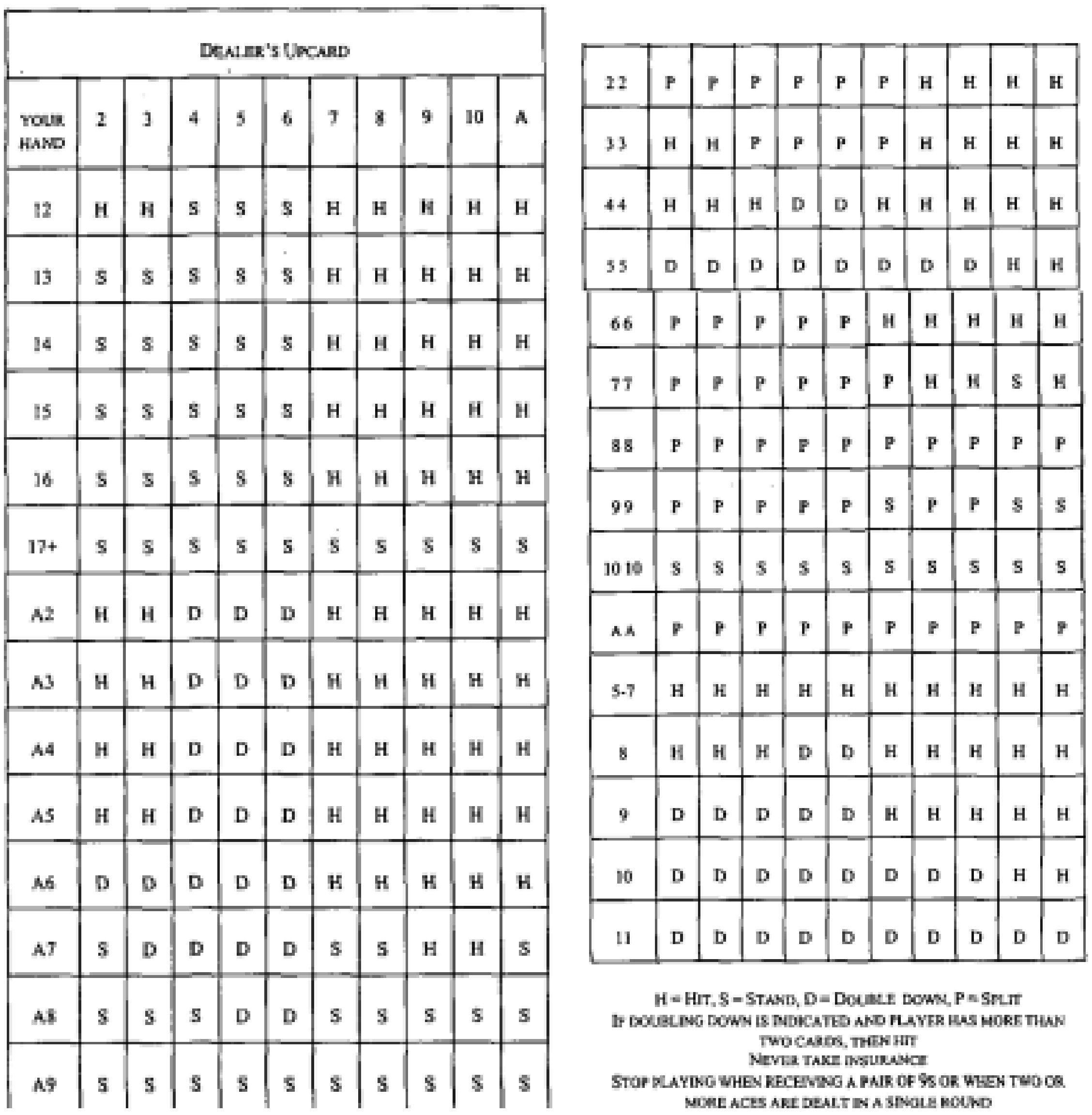
The gaming console contains different predefined and user-defined functions for calculating the value of hands, storing the value of cards, comparing the scores of dealer and player, etc.



```
Dealer score: 22 Cards: 7S 5D KS
Player1 score: 19 Cards: 8C AS
Player1 won!
```

All the functions mentioned above have been used in the output image, shown above. The 4 card suits have been declared by their first letter in the program.

For complexity calculations, n is taken as the number of players.



	Dealer's Upcard									
Your Hand	2	3	4	5	6	7	8	9	10	A
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	H	H
16	S	S	S	S	S	H	H	H	H	H
17+	S	S	S	S	S	S	S	S	S	S
A2	H	H	D	D	D	H	H	H	H	H
A3	H	H	D	D	D	H	H	H	H	H
A4	H	H	D	D	D	H	H	H	H	H
A5	H	H	D	D	D	H	H	H	H	H
A6	D	D	D	D	D	H	H	H	H	H
A7	S	D	D	D	D	S	S	H	H	S
A8	S	S	S	D	D	S	S	S	S	S
A9	S	S	S	S	S	S	S	S	S	S
2 2	P	P	P	P	P	P	H	H	H	H
3 3	H	H	P	P	P	P	H	H	H	H
4 4	H	H	H	D	D	H	H	H	H	H
5 5	D	D	D	D	D	D	D	D	H	H
6 6	P	P	P	P	P	H	H	H	H	H
7 7	P	P	P	P	P	P	H	H	S	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
10 10	S	S	S	S	S	S	S	S	S	S
A A	P	P	P	P	P	P	P	P	P	P
5-7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	D	D	H	H	H	H	H
9	D	D	D	D	D	H	H	H	H	H
10	D	D	D	D	D	D	D	D	H	H
11	D	D	D	D	D	D	D	D	D	D

H = Hit, S = Stand, D = Double down, P = Split
If doubling down is indicated and player has more than two cards, then hit
Never take insurance
Stop playing when receiving a pair of 9s or when two or more aces are dealt in a single round

Moves according to Thorp's strategy used in the program.

4) Implementation and Result :

The play() function with Thorp's Strategy and other functions defined within it given below-

```
void play(vector<player> &players){
   time_t start,stop,start1,end;
   time( &start );
   char input; //for our input

   for(int i =1;i<players.size();i++){
            bet(players[i]); //players bet here
            cout << "Money: " << players[i].info.money << setw(10) << "Bet: " << players[1].bet << endl;
   }


   for(int i = 0;i< (players.size()*2);i++){
            players[(i%players.size())].hand.push_back(deal());
            if((i%players.size()) == 0 && (i/2) == 0){ //the dealers first card
                        players[(i%players.size())].hand[(i%2)].up = false; //is set to false since it's face down
            }
   }
   /**
            The below function shows each players score but the dealers
   */
   for(int i=1;i<players.size();i++){
            cout << players[i].info.username << " hand value is: " << score(players[i].hand) << setw(10) << endl;
   }

   /**
            The below function displays each persons cards
   */
   for(int i =0;i<players.size();i++){
            cout << players[i].info.username << " Cards:" << endl;
            printCards(players[i].hand);
   }


   bool cont = true;
   for(int i = 1;i<players.size();i++){
            do{
                        if(players[0].hand[1].value == 1 && cont){ //if the dealer has an ace and cont is true - basically this
only happens the first time if the dealer doesn't have blackjack
                                    insurance(players);
                                    if(score(players[0].hand) == 21){ //checks to see if the dealer has blackjack - we know the
first card is an A
                                                players[0].hand[0].up = true; //if they do we set the first card to face up
```

```
                printCards(players[0].hand); //prints the dealers cards


                for(int i =1;i<players.size();i++){
                    payout(players[0],players[i]);//to deduct the bet from the money
                }
                input = 'S'; //sets input to stay since they just lost
            }
            cont = false; //if the dealer didn't have blackjack this is now false
        }
        if(players[0].hand[1].value >= 10 && cont){ //if the dealer has a 10 or face card showing we don't
check for insurance but if they have blackjack that's game
            if(score(players[0].hand) == 21){ //if they have blackjack
                players[0].hand[0].up = true; //puts the dealers first card face up

                printCards(players[0].hand); //prints the dealers card
                /**
                    The below function pays out the players since they just lost
                */
                for(int i =1;i<players.size();i++){
                    payout(players[0],players[i]);
                }
                input = 'S'; //input is now S since the players lost
            }
            cont = false; //if the dealer doesn't have 21 we don't care about this anymore
        }
        /**
            As long as the players score is less than 21
        */
        if(score(players[0].hand) <= 21){
                if(score(players[i].hand)<=11 && !hasAce(players[i].hand) &&
(players[i].hand[0].value != players[i].hand[1].value))
                {
                    cout << players[i].info.username << " score is : " <<
score(players[i].hand) << endl;
                    cout << "Would you like to Double Down(D) or take a hit(H) default is to
take a stay?" << endl;
                    if(score(players[i].hand)<=7) cout<<"You must take a hit(H) according
to the strategy";
                    else if(score(players[i].hand) == 8)
                    {
                        if(players[0].hand[1].value<4 || (players[0].hand[1].value<10
&& players[0].hand[1].value>6)) cout<<"You must take a hit(H) according to the strategy ";
                        else cout<<"You must do Double DOWN(D) according to the
strategy ";
                    }
                    else if(score(players[i].hand) == 9)
                    {
                        if((players[0].hand[1].value<10&&players[0].hand[1].value>6) ||
players[0].hand[1].value==1) cout<<"You must take a hit(H) according to the strategy ";
```

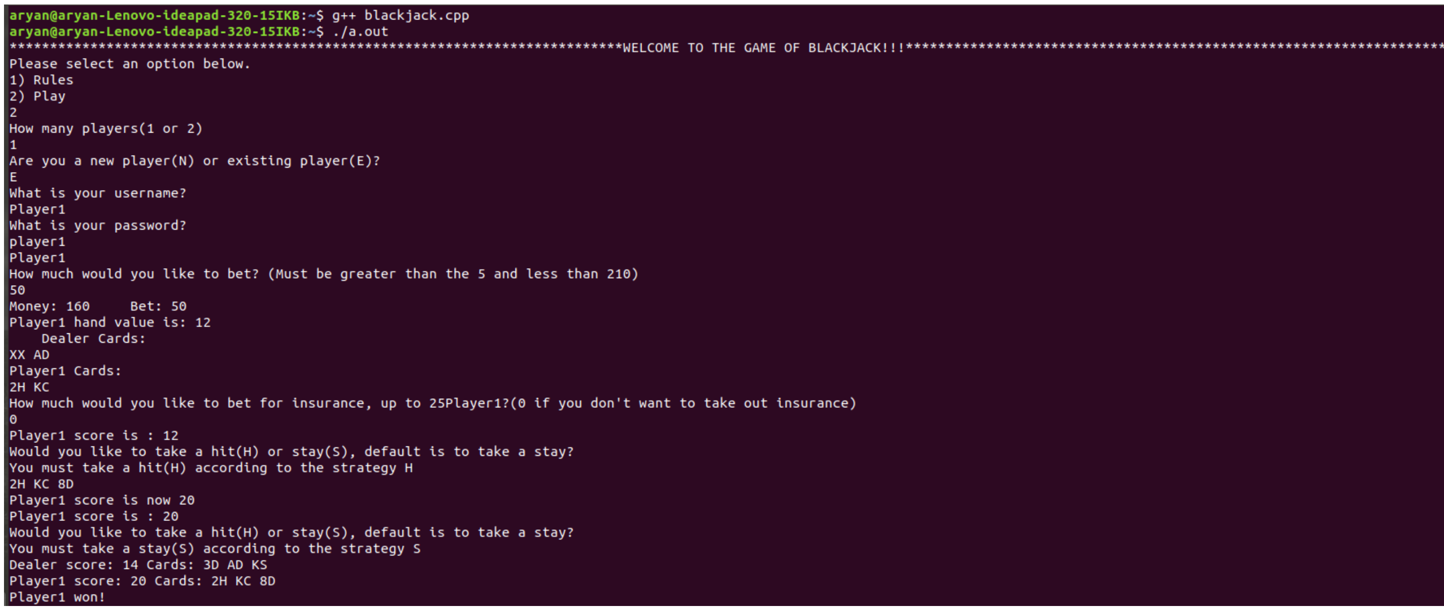
```
								else cout<<"You must do Double DOWN(D) according to the 
strategy ";
							}
							else if(score(players[i].hand) == 10)
							{
								if(players[0].hand[1].value == 1 || 
players[0].hand[1].value==10) cout<<"You must take a hit(H) according to the strategy ";
								else cout<<"You must do Double DOWN(D) according to the 
strategy ";
							}
							else cout<<"You must do Double DOWN(D) according to the strategy ";
						}
						else if((score(players[i].hand)<=20&&score(players[i].hand)>=12) && 
!hasAce(players[i].hand) && (players[i].hand[0].value != players[i].hand[1].value))
						{
							cout << players[i].info.username << " score is : " << 
score(players[i].hand) << endl;
							cout << "Would you like to take a hit(H) or stay(S), default is to take a 
stay?" << endl;
							if(score(players[i].hand)==12)
							{
								if(players[0].hand[1].value<=6 && 
players[0].hand[1].value>=4) cout<<"You must take a stay(S) according to the strategy ";
								else cout<<"You must take a hit(H) according to the strategy ";
							}
							else if(score(players[i].hand)<=16 && score(players[i].hand) > 12)
							{
								if(players[0].hand[1].value<=6 && 
players[0].hand[1].value>=2) cout<<"You must take a stay(S) according to the strategy ";
								else cout<<"You must take a hit(H) according to the strategy ";
							}
							else  cout<<"You must take a stay(S) according to the strategy ";
						}
						else if((score(players[i].hand)<=20&&score(players[i].hand)>=12) && 
hasAce(players[i].hand) && (players[i].hand[0].value != players[i].hand[1].value))
						{
							cout << players[i].info.username << " score is : " << 
score(players[i].hand) << endl;
							cout << "Would you like to take a hit(H), Double Down(D) or stay(S), 
default is to take a stay?" << endl;
							if(score(players[i].hand)>=13 && score(players[i].hand)<=16)
							{
								if(players[0].hand[1].value<=6 && 
players[0].hand[1].value>=4) cout<<"You must make a Double Down(D) according to the strategy ";
								else cout<<"You must take a hit(S) according to the strategy ";
							}
							else if(score(players[i].hand) == 17)
							{
								if(players[0].hand[1].value<=6 && 
players[0].hand[1].value>=2) cout<<"You must make a Double Down(D) according to the strategy ";
								else cout<<"You must take a hit(S) according to the strategy ";
							}
							else if(score(players[i].hand) == 18)
							{
```

```
                                        if(players[0].hand[1].value>=3&&players[0].hand[1].value<=6)
cout<<"You must make a Double Down(D) according to the strategy ";
                                        else
if(players[0].hand[1].value>=9&&players[0].hand[1].value<=10) cout<<"You must take a hit(H) according to the
strategy ";
                                        else "You must take a stay(S) according to the strategy ";
                                }
                                else if(score(players[i].hand) == 19)
                                {
                                        if(players[0].hand[1].value>=5&&players[0].hand[1].value<=6)
cout<<"You must make a Double Down(D) according to the strategy ";
                                        else cout<<"You must take a stay(S) according to the strategy
";
                                }
                                else cout<<"You must take a stay(S) according to the strategy ";
                        }
                        else if((score(players[i].hand)<=20&&score(players[i].hand)>=4) &&
(players[i].hand[0].value == players[i].hand[1].value))
                        {
                                cout << players[i].info.username << " score is : " <<
score(players[i].hand) << endl;
                                cout << "Would you like to take a hit(H), Double Down(D), split(L) or
stay(S), default is to take a stay?" << endl;
                                if(players[i].hand[0].value == 2&&players[i].hand[1].value == 2)
                                {
                                        if(players[0].hand[1].value<=7 &&
players[0].hand[1].value>=2) cout<<"You must split(L) according to the strategy ";
                                        else cout<<"You must take a hit(H) according to the strategy
";
                                }
                                else if(players[i].hand[0].value == 3&&players[i].hand[1].value == 3)
                                {
                                        if(players[0].hand[1].value<=7 &&
players[0].hand[1].value>=4) cout<<"You must split(L) according to the strategy ";
                                        else cout<<"You must take a hit(H) according to the strategy
";
                                }
                                else if(players[i].hand[0].value == 4&&players[i].hand[1].value == 4)
                                {
                                        if(players[0].hand[1].value<=6 &&
players[0].hand[1].value>=5) cout<<"You must make a Double Down(D) according to the strategy ";
                                        else cout<<"You must take a hit(H) according to the strategy
";
                                }
                                else if(players[i].hand[0].value == 5&&players[i].hand[1].value == 5)
                                {
                                        if(players[0].hand[1].value<=9 &&
players[0].hand[1].value>=2) cout<<"You must make a Double Down(D) according to the strategy ";
                                        else cout<<"You must take a hit(H) according to the strategy
";
                                }
                                else if(players[i].hand[0].value == 6&&players[i].hand[1].value == 6)
                                {
                                        if(players[0].hand[1].value<=6 &&
players[0].hand[1].value>=2) cout<<"You must split(L) according to the strategy ";
```

```
                                        else cout<<"You must take a hit(H) according to the strategy
";
                                }
                                else if(players[i].hand[0].value == 7&&players[i].hand[1].value == 7)
                                {
                                        if(players[0].hand[1].value<=7 &&
players[0].hand[1].value>=2) cout<<"You must split(L) according to the strategy ";
                                        else if(players[0].hand[1].value<=10) cout<<"You must take a
stay(S) according to the strategy ";
                                        else cout<<"You must take a hit(H) according to the strategy
";
                                }
                                else if(players[i].hand[0].value == 8&&players[i].hand[1].value == 8)
cout<<"You must split(L) according to the strategy ";
                                else if(players[i].hand[0].value == 9&&players[i].hand[1].value == 9)
                                {
                                        if((players[0].hand[1].value<=6 &&
players[0].hand[1].value>=2)||(players[0].hand[1].value<=9 && players[0].hand[1].value>=8))
                                        cout<<"You must split(L) according to the strategy ";
                                        else
                                        cout<<"You must take a stay(S) according to the strategy ";
                                }
                                else if(players[i].hand[0].value == 10&&players[i].hand[1].value == 10)
                                {
                                        cout<<"You must take a stay(S) according to the strategy ";
                                }
                                else if(players[i].hand[0].value == 1&&players[i].hand[1].value == 1)
                                cout<<"You must split(L) according to the strategy ";
                        }time(&stop);
                        cin >> input; //takes in the input
                        time(&start1);
                        switch(input){ //what did they choose?
                        case 'L': //they wanted to split
                                split(players[0], players[i]); //we split them
                                printCards(players[i].hand); //reprint their cards in case they forgot
                                break;
                        case 'D':
                                doubleDown(players[0], players[i]); //they double down
                                input = 'S'; //sets input to S since now they are done
                                break;
                        case 'H':
                                players[i].hand.push_back(hitMe()); //we give them one more card for
their hit
                                printCards(players[i].hand); //reprint their cards
                                cout << players[i].info.username << " score is now " <<
score(players[i].hand) << endl; //reprint their score
                                break;
                        default: //this is here for people can't follow directions
                                input = 's'; //input is S
                        }
                        if(score(players[i].hand) > 21){ //if they bust they are done
                                input = 'S'; //so we can quit
```

```
                                                }
                        }
            }while(!(input == 'S' || input == 's'));
    }

    dealer_play(players[0]);

    players[0].hand[0].up = true;

    /**
            The below method shows everybody's score and cards including dealers
    */
    for(int i =0;i<players.size();i++){
            cout << players[i].info.username << " score: " << score(players[i].hand) << " Cards: ";
            printCards(players[i].hand);
    }


    for(int i =1;i<players.size();i++){
            if(score(players[i].hand) > 21){ //if the player busted we tell them
                    cout << "You busted! ";
            }
            int win = winner(players[0], players[i]); //we figure out who wins
            if(win == 1){
                    players[i].info.wins += 1; //if the player wins we add one to their win record
            }
            payout(players[0],players[i]); //we payout everybody
            clear(players[i].hand); //we clear out their hands
            players[i].info.total_played+=1; //adds one to the total played
    }

    clear(players[0].hand); //clear out the dealers hand
            time(&end);
    double time_taken = double(end - start1 + stop - start);
            cout << "Time taken by program is : " << fixed << time_taken << setprecision(8);
            cout << " sec " << endl;
}
```

OUTPUT(for a single player game) -



```
aryan@aryan-Lenovo-ideapad-320-15IKB:~$ g++ blackjack.cpp
aryan@aryan-Lenovo-ideapad-320-15IKB:~$ ./a.out
****************************************************************************WELCOME TO THE GAME OF BLACKJACK!!!*******************************************************************
Please select an option below.
1) Rules
2) Play
2
How many players(1 or 2)
1
Are you a new player(N) or existing player(E)?
E
What is your username?
Player1
What is your password?
player1
Player1
How much would you like to bet? (Must be greater than the 5 and less than 210)
50
Money: 160     Bet: 50
Player1 hand value is: 12
    Dealer Cards:
XX AD
Player1 Cards:
2H KC
How much would you like to bet for insurance, up to 25Player1?(0 if you don't want to take out insurance)
0
Player1 score is : 12
Would you like to take a hit(H) or stay(S), default is to take a stay?
You must take a hit(H) according to the strategy H
2H KC 8D
Player1 score is now 20
Player1 score is : 20
Would you like to take a hit(H) or stay(S), default is to take a stay?
You must take a stay(S) according to the strategy S
Dealer score: 14 Cards: 3D AD KS
Player1 score: 20 Cards: 2H KC 8D
Player1 won!
```

Complexity- We have used n as the number of players for calculating the complexity of the play() function.

Alternative- We can also use the number of cards in the hand as n but the number of cards in the hand will be limited as the value of hand can’t exceed 21. Whereas, the number of players is user entered.

The complexity of the program is O(n).

Execution time of the play() function-The time taken for the play() function for n=1 player is 5.00 sec, for n=2 players is 20.00 sec.

Conclusion-

-> In the game, the decisions made during the course of play can affect the probabilities of being in different states in future plays, so it is reasonable to view the challenge of finding optimum strategies as a nonlinear problem that should be addressed by lifelike simulation of sequences of hands played until a deck or decks is/are reshuffled.

-> Results from unpublished experiments performed many years ago indicate that strategies can be evolved using simple variation and selection procedures that outperform some published strategies found in popular books.

Code repository-

The code,presentation and report have been uploaded to the url- https://github.com/Prashant-1108/blackjack.git

References-

-> https://ieeexplore.ieee.org/document/1331064
-> https://ieeexplore.ieee.org/document/1299399