PROJECT REPORT

Topic: The 21 Card Game

UCT 401 Operating Systems

Submitted by:
Ayush Anand 101918060
Akshat Bhatia 101918076

CSBS-3

Submitted to: Rupali Bhardwaj

OBJECTIVE

Our project is a card game that is inspired by the game BlackJack. The goal is to get 21, the best score, or less by adding up the values of cards drawn. Numbered cards have the value of the number on the card, face cards (Jack, Queen, and King) have a value of 11,12, and 13 respectively. The Ace card has a value of 1.
A person can win in 3 possible ways

1. If a player gets a perfect score first, that is 21 he/she wins.
2. If both players' sum of cards value is less than 21, then the higher value of cards wins.
3. If a player's value of cards becomes greater than 21 then the opponent wins.

SCOPE

The game starts with dealing the 2 players with a single card each from a shuffled virtual deck of 52 cards, and the sum of both the players is calculated.

Round 1 is commenced

1. The game asks Player 1 if he/she wants to draw another card from the deck
2. If he says YES, Player 1 gets another card and the chance is passed to Player 2. If he says NO, Player 1 would not be allowed to play in any subsequent rounds
3. If Player 1's sum of cards exceeds 21. He automatically loses.
4. Now it is Player 2's turn, he is given the same option as Player 1, with the same condition
5. If Player 2's sum of cards exceeds 21. He automatically loses.

Round 1 Ends

Round 2 is commenced

1. The same choice is given to Player 1 if he chooses to not pick a card, he would not be able to pick a card in any subsequent rounds.
2. The same choice is repeated for Player 2.
3. If both players choose to not draw a card in any round, then the decision depends on the sum of cards.

Round 2 Ends

After comparing the sum of cards, the player with a higher value of cards, WINS THE GAME

The scoreboard is updated for the players

At any given point in time, out of the 52 cards, No card can be repeated in a single game

METHODOLOGY

1. Our game starts by asking players an option to either play or quit. - This is a menu-driven function that will run every time after a game is finished until any player wishes to quit. Here we are also updating the points of both the players to get a win after a series of games are played.

2. After every game, we are using the ‘reset’ function to
    a. Clear the match details of the previous game.
    b. Reset the array of cards.
    c. Reset the sum of card values of each player(where the total sum of the card is stored)
    d. Reset the card value array of each player. (where values of all the drawn card of a player are stored)

3. Once the game starts both players are assigned a random card from the deck of shuffled cards. We are storing the card number in an array. We are using the ‘rand’ function to assign random cards to each player. No two same cards can be drawn in a single game. Every time a player draws a card, that particular card gets deleted from the array to ensure it is never repeated. This iterative process of drawing cards continues until a player passes his turn or he wins or loses by any criteria mentioned above. If a player passes his turn then that player’s flag gets reset and he/she will never be able to draw a card again in that game. After each draw, the sum of card values is stored to check the winning conditions. As soon as the sum of card values reaches greater than 21 for any player, that particular player gets busted and the code terminates instantly after which a “winner” message is displayed to the winning player.

4. All the card values drawn by a player are stored in their separate arrays to calculate the total value of cards that a player has.

FUTURE SCOPE

This single card game has major scope for future

1. It can be converted into a single-player game i.e. playing against the computer, or a multiplayer game, i.e. playing with 3 or people
2. Using shell script, effects and animations can be added to the game to make it more user-friendly and engaging for the players.
3. This existing game can have multiple variations and give users multiple ways to enjoy the same game, for example,
    a. Adding a Joker card, that can take on any value during the game
    b. Adding a component to bet money, to increase the risk and reward
    c. Letting the cards repeat themselves, to increase the suspense in the game, etc

SOURCE CODE

```
#!/usr/bin/bash

cards=( 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7
8 8 8 8 9 9 9 9 10 10 10 10 11 11 11 11 12 12 12 12 13 13 13 13
)

player1=()
declare -a player2

flag=0

player1Play=1
player2Play=1

total1=0
total2=0

winner=0
player1Points=0
player2Points=0
```

```
reset()
{
    cards=( 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7
7 7 8 8 8 8 9 9 9 9 10 10 10 10 11 11 11 11 12 12 12 12 13 13 13
13 )

    player1=()
    player2=()

    flag=0

    player1Play=1
    player2Play=1

    total1=0
    total2=0

    winner=0
    clear
}

addToDeck()
{
    if [ $flag -eq 0 ]
    then
        total1=0
        player1+=( "$num" )
        echo -n "Player 1 Cards:"
        echo "${player1[@]}"
        for i in ${player1[*]}
        do
            total1=`expr $total1 + $i`
        done
        echo -n "Sum of Card Values:"
        echo $total1
        echo "~~~~~~~~~~~~~~~~~"
    else
        total2=0
        player2+=( "$num" )
        echo -n "Player 2 Cards:"
```

```
        echo "${player2[@]}"
        for i in ${player2[*]}
        do
            total2=`expr $total2 + $i`
        done
        echo -n "Sum of Card Values:"
        echo $total2
        echo "~~~~~~~~~~~~~~~~~"
    fi
}

getWinner()
{
    if [ $total1 -gt $total2 ]
    then
        winner="Player1"
    else
        winner="Player2"
    fi
}

newGame()
{
    echo "LET'S START"
    echo " "
    rand=$[$RANDOM % ${#cards[@]}]
    echo "~~~~~~~~~~~~~~~~~"
    num=${cards[$rand]}
    addToDeck num flag
    echo " "
    flag=1
    rand=$[$RANDOM % ${#cards[@]}]
    echo "~~~~~~~~~~~~~~~~~"
    num=${cards[$rand]}
    addToDeck num flag
    flag=0
    del_element=`expr $rand + 1`
    cards=("${cards[@]:0:$((del_element-1))}"
"${cards[@]:$del_element}")
}
```

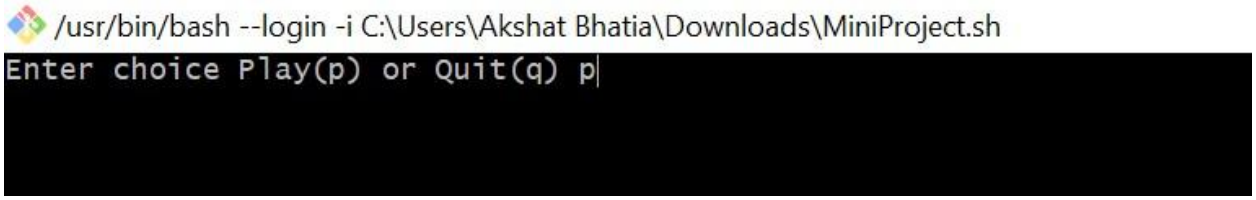
```
playGame()
{
    reset
    newGame
    while [ $player1Play -eq 1 ] || [ $player2Play -eq 1 ]
    do
    echo " "
    if [ $flag -eq 0 ]
    then
        echo "Player 1's Chance"
        echo "Current total: $total1"
    else
        echo "Player 2's Chance"
        echo "Current total: $total2"
    fi

    echo -n "Draw another card or Pass (y or n):"
    read draw
    if [ $draw == "y" ]
    then
        rand=$[$RANDOM % ${#cards[@]}]
        echo "~~~~~~~~~~~~~~~~~"
        num=${cards[$rand]}
        if [ $flag -eq 0 ] && [ $player1Play -eq 1 ]
        then
            addToDeck num flag
            if [ $player2Play -eq 1 ]
            then
                flag=1
            fi
        elif [ $flag -eq 1 ] && [ $player2Play -eq 1 ]
        then
            addToDeck num flag
            if [ $player1Play -eq 1 ]
            then
                flag=0
            fi
        fi
        del_element=`expr $rand + 1`
        cards=("${cards[@]:0:$((del_element-1))}"
"${cards[@]:$del_element}")
```

```
    if [ $total1 -eq 21 ]
    then
        winner="Player1"
        break
    fi

    if [ $total2 -eq 21 ]
    then
        winner="Player2"
        break
    fi



    if [ $total1 -gt 21 ]
    then
        winner="Player2"
        clear
        echo "It is a BUST for Player 1"
        sleep 3s
        break
    fi

    if [ $total2 -gt 21 ]
    then
        winner="Player1"
        clear
        echo "It is a BUST for Player 2"
        sleep 3s
        break
    fi



elif [ $flag -eq 0 ] && [ $draw == "n" ]
then
     player1Play=0
     flag=1
```

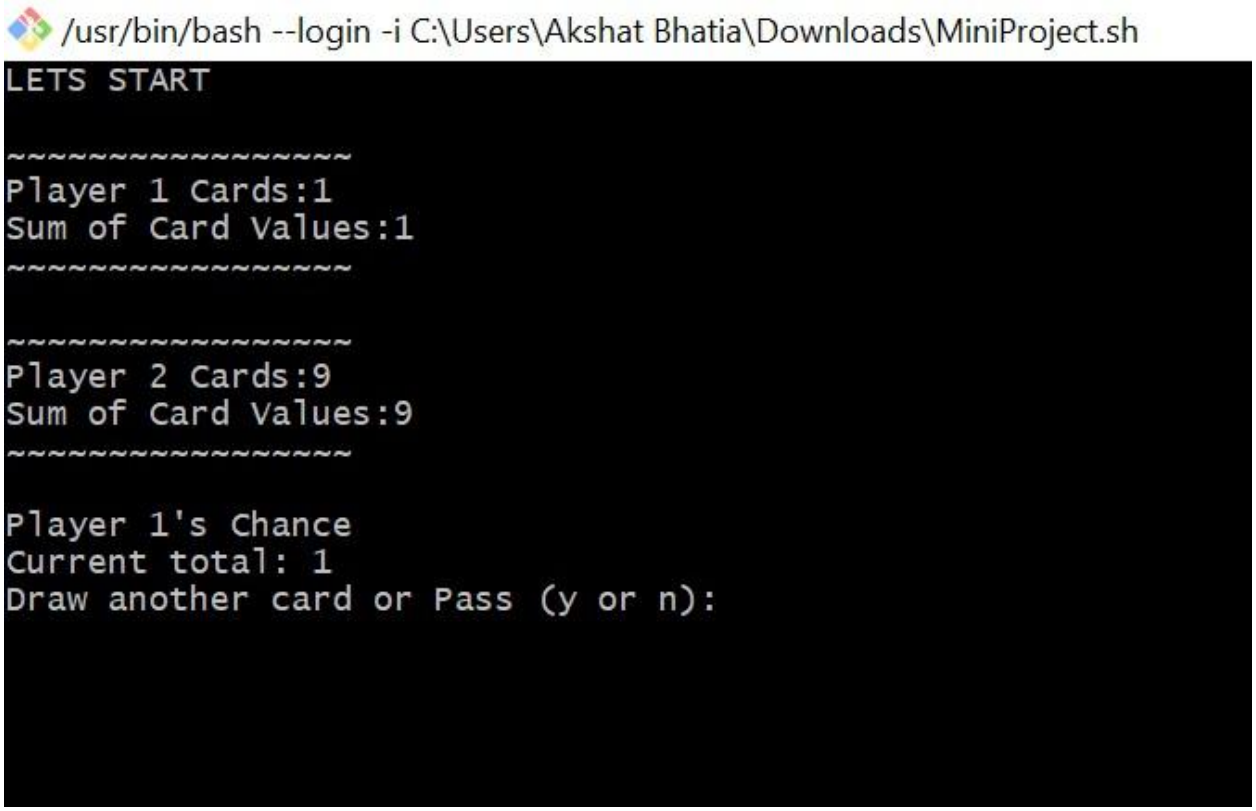
```
     elif [ $flag -eq 1 ] && [ $draw == "n" ]
     then
         player2Play=0
         flag=0
     fi
     getWinner
    done
}

while true
do
echo -n "Enter choice Play(p) or Quit(q) "
read ch
    case "$ch" in
        p) playGame
        ;;
        q) break
    esac
clear
if [ $winner == "Player1" ]
then
    echo "Player 1 WON!!"
    player1Points=`expr $player1Points + 1`
elif [ $winner == "Player2" ]
then
    echo "Player 2 WON!!"
    player2Points=`expr $player2Points + 1`
fi
echo "Player 1 points: $player1Points"
echo "Player 2 points: $player2Points"
done
```

OUTPUT

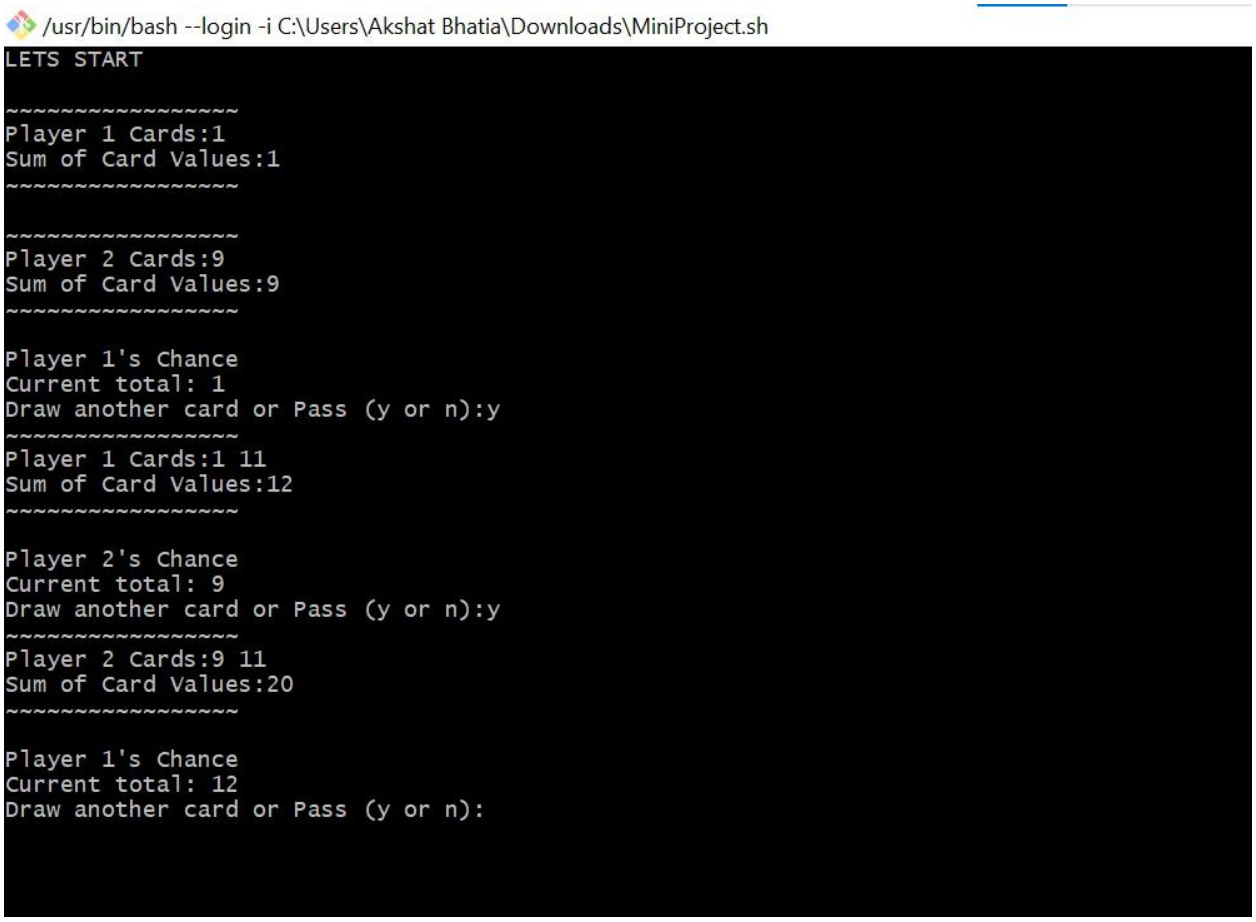
GAME STARTS



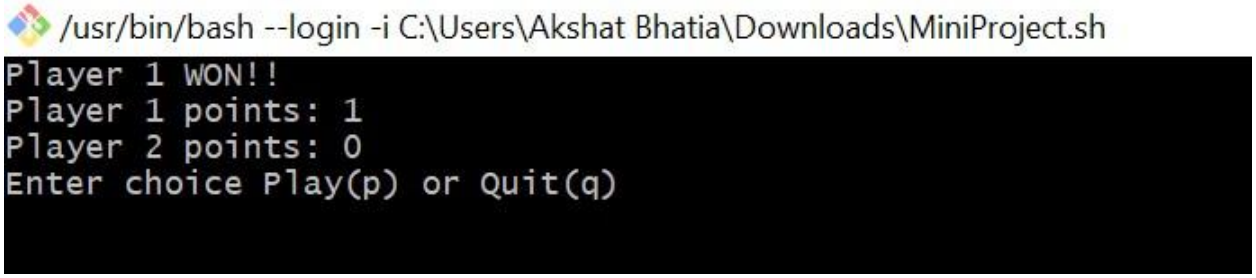
EACH PLAYER GETS 1 RANDOM CARD



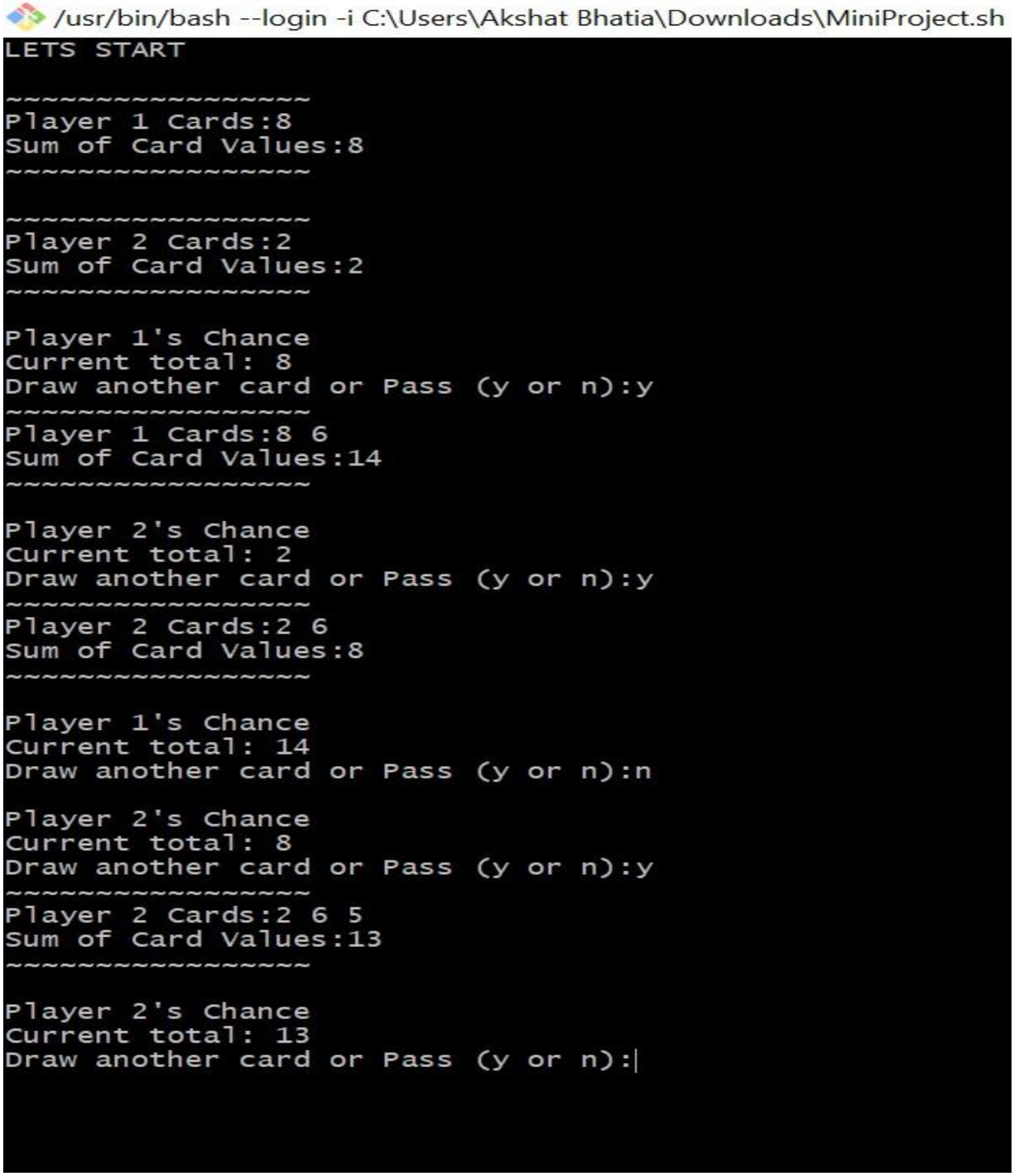
THE ENTIRE GAMEPLAY



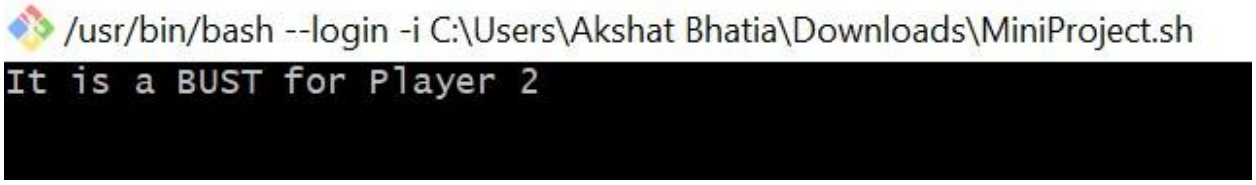
POINTS AFTER ROUND 1



ROUND 2 BEGINS



PLAYER 2 GETS BUST, AS VALUE OVER 21



POINTS AFTER ROUND 2