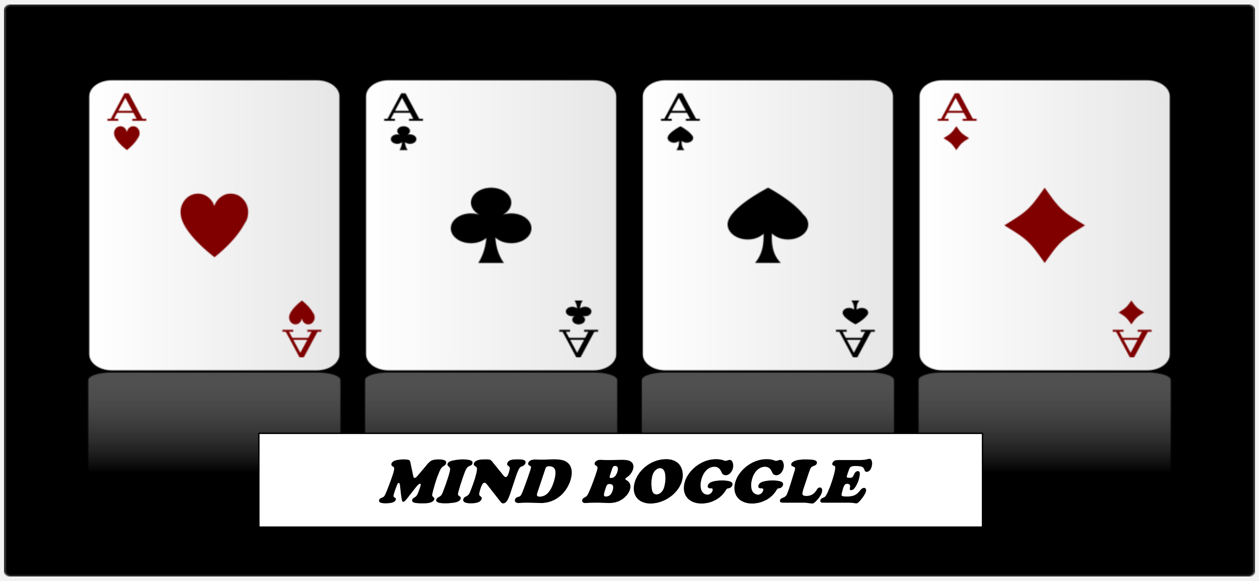
OBJECT ORIENTED PROGRAMMING [CT-260]

PROJECT REPORT



GROUP MEMBERS :

MUHAMMAD FURQAN PATEL	CR-032
ANAS QUTBI	CR-037
ABUZAR HASHMI	CR-042

1. PROBLEM PLANNING:

How to design a unique card game that is intuitive and engaging for the human player.
How to implement the logic and rules of the game, such as the special effects of certain cards, the scoring system, and the end condition.
How to program the bots to have different levels of difficulty and personality, such as being aggressive, cautious, or unpredictable.
How to test and debug your code to ensure it works correctly and smoothly.

2. PROBLEM DISCRIPTION:

Mind Boggle: Cards Edition is a card game where players compete against each other and computer-controlled opponents (bots) to achieve the lowest score. The game involves strategic decision-making and memory skills. Each player is dealt a hand of three cards from a deck of 52 cards.

The objective of the game is to obtain the highest score by strategically discarding cards and exchanging them with other players' cards. Players can discard cards by either choosing to replace their own card with a new card from the deck or discarding a card without replacement. Certain cards have special effects that can impact the game, such as shuffling opponents' cards, exchanging cards with other players, or revealing the player's own cards.

The game features a human player and three computer-controlled bots. The bots have a memory mechanism that allows them to remember the rank, condition, and position of their own and opponents' cards. The bots use this memory to make decisions about discarding and exchanging cards. The human player can make choices by selecting the rank and suit of the card to be discarded or exchanged.

At the end of the game, the scores of all players are calculated based on the ranks of the cards in their hands. The player with the lowest score is declared the winner.

3. FEATURES OF OUR PROGRAM:

- Human player and three computer-controlled bots
- Strategic decision-making and memory skills
- Discard cards with or without replacement
- Score calculation and determination of the winner.

4. MOST CHALLENGING PART:

- **UNDERSTANDING THE FLOW:**

The code involves a game implementation that includes multiple classes, functions, and memory management. Understanding the overall flow of the game might be challenging, especially for someone unfamiliar with the experience of working with that many values.

- **MEMORY MANAGEMENT AND TRACKING**:

The code includes memory tracking for both players and bots. Keeping track of the cards in memory and updating the memory based on player moves and bot decisions can be complex.

- **DECISION-MAKING FOR BOTS:**

The code contains logic for bot moves, including decision-making based on memory and card conditions. Implementing effective and strategic decision-making algorithms for bots can be challenging, especially to create competitive and intelligent opponents.

- **HANDLING USER INPUT AND INTERACTION:**

The code includes user input for the player's moves and interaction. Ensuring proper input validation, handling unexpected input, and providing appropriate feedback to the user can be challenging.

- **OVERALL CODE COMPLEXITY AND DEBUGGING:**

The code includes various classes, functions, loops, and conditional statements. Managing and debugging complex code can be challenging, especially when trying to identify and fix any potential issues or bugs.

5. INDIVIDUAL CONTRIBUTIONS:

Most of the work was done in group but some individual contributions are:

1. MUHAMMAD FURQAN PATEL [CR-032]

- Syntax of the code.
- Project report.

2. ANAS QUTBI [CR-037]

- Identify the bugs in the code and debug them.
- Help in creating project report.

3. ABUZAR HASHMI [CR-042]

- Main idea of the project.
- UML of the code.

6. FUTURE EXPANSION:

MULTIPLAYER SUPPORT: Currently, the code supports a single human player against a bot player. A future expansion could involve adding support for multiple human players, allowing them to play against each other or against multiple bot players. This would require modifications to the game logic, such as handling turns, managing player scores, and determining the overall winner.

ADDITIONAL GAME MODES: The code currently implements a basic card game with predefined rules. A future expansion could involve adding new game modes or variations to make the game more interesting. For example, different types of card games or custom rulesets could be implemented, allowing players to choose their preferred game mode.

USER INTERFACE (UI) ENHANCEMENTS: The code primarily relies on console output for displaying game information. A future expansion could involve developing a graphical user interface (GUI) to provide a more visually appealing and interactive experience. This could include features such as card animations, player avatars, and intuitive controls for gameplay interactions.

7. OOP CONCEPTS USED DURING WORKING ON THE PROJECT(CODE BREAKDOWN):

CLASS INHERITANCE: The code demonstrates class inheritance by creating the Human and Bot classes derived from the Player class. This shows how derived classes can inherit properties and methods from a base class.

POLYMORPHISM: The code utilizes polymorphism through function overriding. The printHand() function is overridden in the Human and Bot classes to provide specialized implementations of the function for each class.

CLASS COMPOSITION: The Game class has member variables that are objects of other classes (Human and Bot). This demonstrates the concept of class composition, where complex classes can be composed of simpler classes.

ENCAPSULATION: The classes in the code encapsulate data and behavior by defining member variables and member functions. Access to these members is controlled through public and private access specifiers.

MEMORY MANAGEMENT: THE code uses dynamic memory allocation to create an array of objects. It dynamically allocates memory for the Card objects and manages the memory allocation and deallocation process.

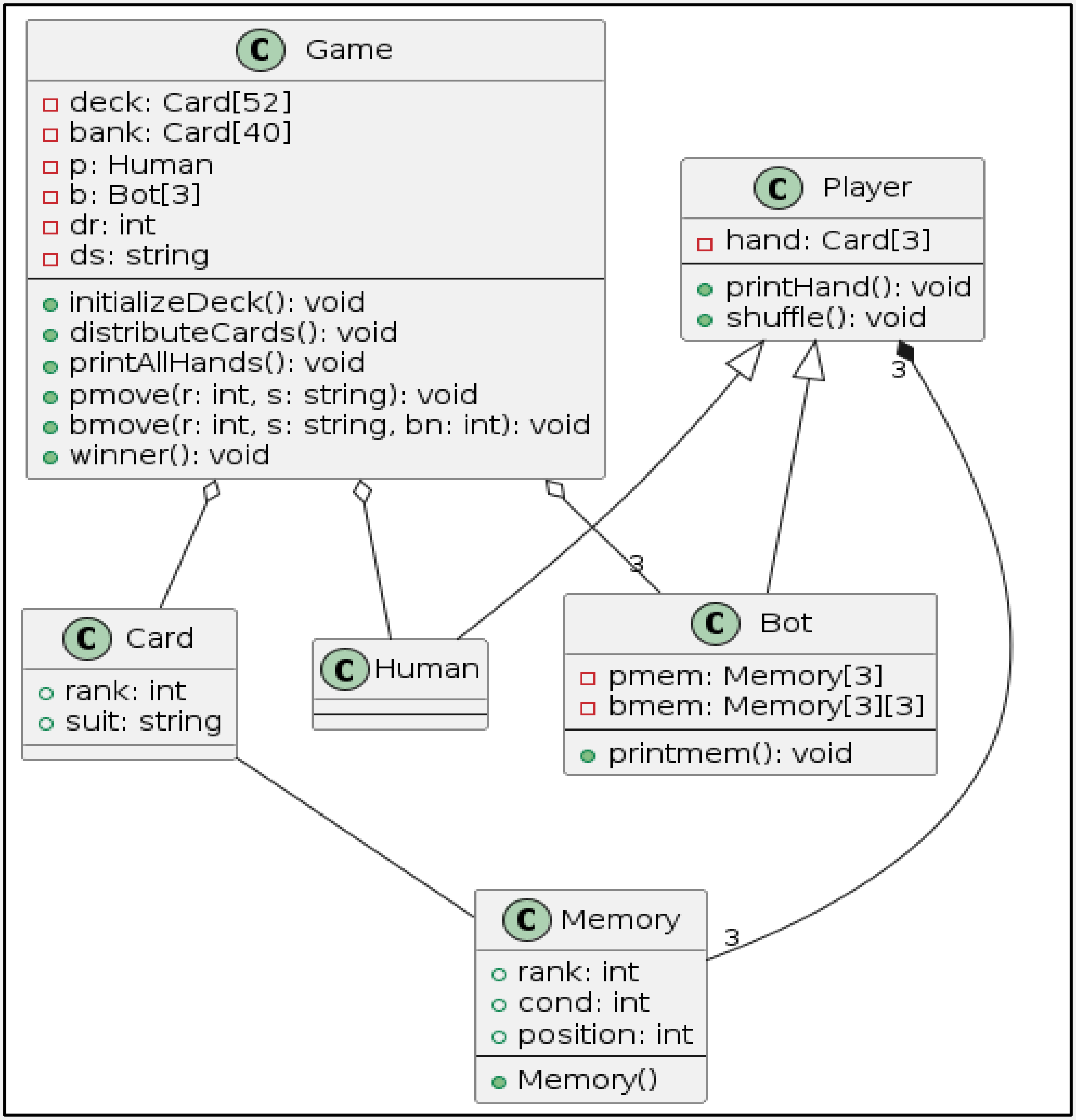
RANDOM NUMBER GENERATION: The code utilizes the rand() function to generate random numbers for shuffling cards, distributing cards, and making decisions during the game. This demonstrates the use of random number generation in a program.

USE OF ARRAYS AND LOOPS: The code employs arrays and loops to store and manipulate multiple objects of the same type. It demonstrates how to iterate over arrays using for loops to perform operations on each element.

GAME LOGIC: The code implements the logic for a card game, including initializing the deck, distributing cards, making moves, shuffling cards, and determining the winner. It shows how OOP concepts can be applied to model and implement game mechanics.

Overall, this code provides practical examples of OOP concepts in action, showcasing how classes, objects, inheritance, polymorphism, and encapsulation can be used to design and implement a real-world scenario.

8. UML DIAGRAM OF THE CODE :



9. SOURCE CODE:

```cpp
#include <iostream>
#include <string>
#include <ctime>
#include <cstdlib>
using namespace std;
class Card {
public:
    int rank;
    string suit;
    };

    class Memory{
    public:
        int rank, cond, position;                    // store rank, condition (known, less than, unknown) and position of own and opponents card
        Memory() {
            rank = 14;                       //default rank = 14 means unknown
            cond = 0;                        // default condition = 0 means unknown
            position = 3;                    //3 means unknown
        }
    };

    class Player {
    public:
        Card hand[3];

        void printHand() {                           //prints card on hand
            for (int i = 0;i < 3;i++) {
                switch (hand[i].rank) {
                case 1:
                    cout << "Ace";
                    break;
                case 11:
                    cout << "Jack";
                    break;
                case 12:
                    cout << "Queen";
                    break;
                case 13:
                    cout << "King";
                    break;
                default:
                    cout << hand[i].rank;
                }
                cout << " of " << hand[i].suit << endl;
            }
        }
        void shuffle() {                            //shuffles card of a hand
            int y = rand() % 3;
            int z = rand() % 3;
            int tempr;
            string temps;
            for (int i = 0;i < 10;i++) {
                tempr = hand[y].rank;
                temps = hand[y].suit;
                hand[y].rank = hand[z].rank;
                hand[y].suit = hand[z].suit;
                hand[z].rank = tempr;
                hand[z].suit = temps;
                y = rand() % 3;
                z = rand() % 3;
            }
        }
    };

    class Human :public Player {
     };

    class Bot:public Player {
    public:
        Memory pmem[3], bmem[3][3];

        void printmem() {                               //prints memory of a bot... func for test only
            cout << "Player cards: " << endl;
            for (int i = 0;i < 3;i++) {
                cout << "Position: " << pmem[i].position << endl << "Rank: " << pmem[i].rank << endl << "Condition: " << pmem[i].cond << endl;
            }
            cout << "Bot Cards:" << endl << endl ;
            for (int i = 0;i < 3;i++) {
                cout << "Bot " << i << endl;
                for (int j = 0;j < 3;j++) {
                    cout << "Position: " << bmem[i][j].position << endl << "Rank: " << bmem[i][j].rank << endl << "Condition: " << bmem[i][j].cond << endl << endl;
                }
            }

        }
    };

    class Game {
    public:
        Card deck[52];
        Card bank[40];
        Human p;
        Bot b[3];
        int dr;                         //discarded rank
        string ds;                      //discarded suit
        void initializeDeck() {                     //intializes deck in sorted order
            int k = 0;
            string s[4] = {"spades","clubs","hearts","diamonds"};
            for (int i = 0;i < 4;i++) {
                for (int j = 1;j < 14;j++) {
                    deck[k].rank = j;
                    deck[k].suit = s[i];
                    k++;
                }
            }
        }

        void distributeCards() {

            int y = rand() % 52;
            for (int i = 0;i < 3;i++) {                 //dist card to player
                while (deck[y].rank == 0) {
                    y = rand() % 52;                    //random num to get card randomly from the deck
                }
                p.hand[i].rank = deck[y].rank;
                p.hand[i].suit = deck[y].suit;
                deck[y].rank = 0;                       //setting value for the rank to 0, since this card is already distributed...
            }
            for (int i = 0;i < 3;i++) {                 //dist card to bots... i is for bot whos recieving card
                for (int j = 0;j < 3;j++) {             // j is for posiiton of card
                    while (deck[y].rank == 0) {
                        y = rand() % 52;
                    }
                    b[i].hand[j].rank = deck[y].rank;
                    b[i].hand[j].suit = deck[y].suit;
                    b[i].bmem[i][j].rank = deck[y].rank;        // bot memorizes its cards here
                    b[i].bmem[i][j].position = j;           //3 means unknown
                    b[i].bmem[i][j].cond = 2;                 //  0 unknown, 1 less than=, 2 known
                    deck[y].rank = 0;
                }
            }

            for (int i = 0;i < 40;i++) {                //distributing remaining cards to bank randomly
                while (deck[y].rank == 0) {
                    y = rand() % 52;
                }
                bank[i].rank = deck[y].rank;
                bank[i].suit = deck[y].suit;
                deck[y].rank = 0;
            }
        }

        void printAllHands() {                              //prints all hands in game... use in end
            cout << endl << "Player Hand" << endl;
            p.printHand();
            for (int i = 0;i < 3;i++) {
                cout << endl << "Bot " << i + 1 << " Hand" << endl;
                b[i].printHand();
            }
        }

        void pmove(int r, string s) {                       //players move
            int m;
            cout << "You picked ";
            switch (r)
            {
            case 1:
                cout << "Ace";
                break;
            case 11:
                cout << "Jack";
                break;
            case 12:
                cout << "Queen";
                break;
            case 13:
                cout << "King";
                break;
            default:
                cout << r;
            }
            cout << " of " << s << endl;
            cout << "Enter -1 to discard or Enter position of your card (0/1/2) to replace it" << endl;
            cin >> m;
            while (m > 2 || m < -1) {
                cout << "Invalid number, enter again" << endl;
            }
            switch (m) {
            case -1:
                dr = r;
                ds = s;
                for (int i = 0;i < 3;i++) {
                    for (int j = 0;j < 3;j++) {
                        if (b[i].pmem[j].rank > dr) {
                            b[i].pmem[j].rank = dr;
```

```
                            b[i].pmem[j].cond = 1;
                        }
                    }
                }
                break;
            default:
                dr = p.hand[m].rank;
                ds = p.hand[m].suit;
                p.hand[m].rank = r;
                p.hand[m].suit = s;

                for (int i = 0;i < 3;i++) {
                    b[i].pmem[m].cond = 1;
                    b[i].pmem[m].position = m;
                    b[i].pmem[m].rank = dr;
                }
            }
            cout << "You Discarded ";
            switch (dr) {
            case 1:
                cout << "Ace";
                break;
            case 11:
                cout << "Jack";
                break;
            case 12:
                cout << "Queen";
                break;
            case 13:
                cout << "King";
                break;
            default:
                cout << dr;
            }
            cout << " of " << ds << endl;
            switch (dr) {
            case 13:                //King thrown.. see your card
            {
                p.shuffle();
                cout << "Your cards are in following order now" << endl;
                p.printHand();
                for (int i = 0;i < 3;i++) {
                    for (int j = 0;j < 3;j++) {
                        b[i].pmem[j].position = 3;
                    }
                }
                break;
            }
            case 10:            //10 ...shuffle oppponent's card
            {
                int x;
                cout << "Enter a number to shuffle bot's cards and view them" << endl;
                cin >> x;
                while (x < 0 || x>2) {
                    cout << "Invalid input, enter the bot number again" << endl;
                    cin >> x;
                }
                b[x].shuffle();
                cout << "Bot " << x << "'s cards are in this order now" << endl;
                b[x].printHand();
                for (int i = 0;i < 3;i++) {
                    for (int j = 0;j < 3;j++) {
                        b[i].bmem[x][j].position = 3;
                    }
                }
                break;
            }
            case 7:             //7... exchange card w/o looking
            {
                int x, y, z;
                cout << "Enter a bot number to exchnage card with" << endl;
                cin >> x;
                while (x < 0 || x>2) {
                    cout << "Invalid input, enter the bot number again" << endl;
                    cin >> x;
                }
                cout << "Enter a card number/positon to exchnage card" << endl;
                cin >> y;
                while (y < 0 || y>2) {
                    cout << "Invalid input, enter the card number again" << endl;
                    cin >> y;
                }
                cout << "Enter your card number/positon to exchnage card" << endl;
                cin >> z;
                while (z < 0 || z>2) {
                    cout << "Invalid input, enter the card number again" << endl;
                    cin >> z;
                }
                int tempr, tempmr, tempmc, tempmp;      //tempmr is for memory rank...
                string temps;
                tempr = p.hand[z].rank;
                temps = p.hand[z].suit;
                p.hand[z].rank = b[x].hand[y].rank;
                p.hand[z].suit = b[x].hand[y].suit;
                b[x].hand[y].rank=tempr;
                b[x].hand[y].suit=temps;
                for (int i = 0;i < 3;i++) {
                    tempmr = b[i].pmem[z].rank;
                    tempmc = b[i].pmem[z].cond;
                    tempmp = b[i].pmem[z].position;
                    b[i].pmem[z].rank=b[i].bmem[x][y].rank;
                    b[i].pmem[z].cond = b[i].bmem[x][y].position;
                    b[i].pmem[z].position = b[i].bmem[x][y].position;
                    b[i].bmem[x][y].rank=tempmr;
                    b[i].bmem[x][y].cond = tempmc;
                    b[i].bmem[x][y].position = tempmp;
                }
                break;
            }
            }
        }

        void bmove(int r, string s, int bn) {
            int pos = 4;
            bool discard_flag = 0;
            int z = rand() % 5;
            int myHighR = 0;
            int myHighC=4;
            if (r <= 6) {
                for (int i = 0;i < 3;i++) {
                    if (b[bn].bmem[bn][i].rank > myHighR && b[bn].bmem[bn][i].cond > 0 && b[bn].bmem[bn][i].position == i) {
                        myHighR = b[bn].bmem[bn][i].rank;
                        myHighC = i;
                        discard_flag = 1;
                    }
                }
                if (discard_flag == 0) {
                    if (z == 0) {
                        goto l1;
                    }
                    for (int i = 0;i < 3;i++) {
                        if (b[bn].bmem[bn][i].rank > myHighR && b[bn].bmem[bn][i].cond > 0 && b[bn].bmem[bn][i].position == 3) {
                            myHighR = b[bn].bmem[bn][i].rank;
                            myHighC = i;
                            discard_flag = 1;
                        }
                    }
                }
                l1:
                if (discard_flag == 1) {
                    if (myHighR < r) {
                        discard_flag = 0;
                    }
                }
                if (discard_flag == 1) {
                    b[bn].hand[myHighC].rank = r;
                    b[bn].hand[myHighC].suit = s;
                    dr = b[bn].hand[myHighC].rank;
                    ds = b[bn].hand[myHighC].suit;
                    b[bn].bmem[bn][myHighC].cond = 2;
                    b[bn].bmem[bn][myHighC].position = myHighC;
                    b[bn].bmem[bn][myHighC].rank = r;
                    pos = myHighC;
                    for (int i = 0;i < 3;i++) {
                        if (i == bn) {
                            continue;
                        }
                        b[i].bmem[bn][myHighC].cond = 1;
                        b[i].bmem[bn][myHighC].position = myHighC;
                        b[i].bmem[bn][myHighC].rank = dr;
                    }
                }
            }
            if (discard_flag == 0) {
                dr = r;
                ds = s;
                for (int i = 0;i < 3;i++) {
                    if (i == bn) {
                        continue;
                    }
                    for (int j = 0;j < 3;j++) {
                        if (b[i].bmem[bn][j].cond == 0) {
                            b[i].bmem[bn][j].cond = 1;
                            b[i].bmem[bn][j].position = j;
                            b[i].bmem[bn][j].rank = dr;
                        }
                    }
                }
            }

            cout << "Bot " << bn << " discarded ";
            switch (dr) {
            case 1:
                cout << "Ace";
                break;
            case 11:
                cout << "Jack";
                break;
            case 12:
                cout << "Queen";
                break;
            case 13:
                cout << "King";
                break;
            default:
                cout << dr;
            }
            cout << " of " << ds;
            switch (pos) {
            case 4: {
                cout << " without exchange" << endl;
                break;
            }
            default:
            {
                cout << " from index " << pos << endl;
            }
            }
            switch (dr) {
            case 13:                            //bots move on king
            {
                b[bn].shuffle();
                for (int j = 0;j < 3;j++) {
                    b[bn].bmem[bn][j].rank = b[bn].hand[j].rank;
                    b[bn].bmem[bn][j].position = j;
                    b[bn].bmem[bn][j].cond = 2;
                }
                for (int i = 0;i < 3;i++) {
                    if (i == bn) {
                        continue;
                    }
                    for (int j = 0;j < 3;j++) {
                        b[i].bmem[bn][j].position = 3;
                    }
                }
                break;
            }
            case 10:                        //bot's move on 10
            {
                int x = 0;
                int z = 0;
                int y = rand() % 4;
                while (x < 6) {
                    while (y == bn) {
                        y = rand() % 4;
                    }
                    switch (y) {
                    case 3:
                    {
                        if ((b[bn].pmem[0].cond > 0 && b[bn].pmem[1].cond > 0 && b[bn].pmem[2].cond > 0) || (b[bn].pmem[0].cond > 0 && b[bn].pmem[1].cond > 0) || (b[bn].pmem[0].cond > 0 && b[bn].pmem[2].cond > 0) || (b[bn].pmem[1].cond > 0 && b[bn].pmem[2].cond > 0)) {
                            break;
                        }
                        p.shuffle();
                        cout << "Bot shuffled your cards" << endl;
                        z = 1;
                        for (int j = 0;j < 3;j++) {
                            b[bn].pmem[j].rank = p.hand[j].rank;
                            b[bn].pmem[j].position = j;
                            b[bn].pmem[j].cond = 2;
                        }
                        for (int i = 0;i < 3;i++) {
                            if (i == bn) {
                                continue;
                            }
                            for (int j = 0;j < 3;j++) {
                                b[i].pmem[j].position = 3;
                            }
                        }
                        break;
                    }
                    default:
                    {
                        if ((b[bn].bmem[y][0].cond > 0 && b[bn].bmem[y][1].cond > 0 && b[bn].bmem[y][2].cond > 0) || (b[bn].bmem[y][0].cond > 0 && b[bn].bmem[y][1].cond > 0) || (b[bn].bmem[y][0].cond > 0 && b[bn].bmem[y][2].cond > 0) || (b[bn].bmem[y][1].cond > 0 && b[bn].bmem[y][2].cond > 0)) {
                            break;
                        }
                        b[y].shuffle();
                        cout << "Bot shuffled cards of bot " << y << endl;
                        z = 1;
                        for (int j = 0;j < 3;j++) {
                            b[bn].bmem[y][j].rank = p.hand[j].rank;
                            b[bn].bmem[y][j].position = j;
                            b[bn].bmem[y][j].cond = 2;
                        }
                        for (int i = 0;i < 3;i++) {
                            if (i == bn) {
                                continue;
                            }
                            for (int j = 0;j < 3;j++) {
                                b[i].bmem[y][j].position = 3;
                            }
                        }
                        break;
                    }

                    }
                    y = rand() % 4;
                    x++;
                }
                if (z== 0) {
                    while (y == bn) {
                        y = rand() % 4;
                    }
                    switch (y) {
                    case 3:
                    {
                        p.shuffle();
                        cout << "Bot shuffled your cards" << endl;
                        z = 1;
                        for (int j = 0;j < 3;j++) {
                            b[bn].pmem[j].rank = p.hand[j].rank;
                            b[bn].pmem[j].position = j;
                            b[bn].pmem[j].cond = 2;
                        }
                        for (int i = 0;i < 3;i++) {
                            if (i == bn) {
                                continue;
                            }
                            for (int j = 0;j < 3;j++) {
                                b[i].pmem[j].position = 3;
                            }
                        }

                        break;
                    }
                    default:
                    {
                        b[y].shuffle();
                        cout << "Bot shuffled cards of bot " << y << endl;
                        z = 1;
                        for (int j = 0;j < 3;j++) {
                            b[bn].bmem[y][j].rank = p.hand[j].rank;
                            b[bn].bmem[y][j].position = j;
                            b[bn].bmem[y][j].cond = 2;
                        }
                        for (int i = 0;i < 3;i++) {
                            if (i == bn) {
                                continue;
                            }
                            for (int j = 0;j < 3;j++) {
                                b[i].bmem[y][j].position = 3;
                            }
                        }
                        break;
                    }
                    }
                }
                break;
            }
            case 7:
            {
                int plow = 4;
                int clow = 14;
                int cLown;
                int myHighR = 0;
                int myHighC = 4;
                bool found = 0;
                for (int i = 0;i < 3;i++) {
                    if (b[bn].pmem[i].rank < clow && b[bn].pmem[i].position == i && b[bn].pmem[i].cond>0) {
                        plow = 3;
                        clow = b[bn].pmem[i].rank;
                        cLown = i;
                        found = 1;
                    }
                }
                for (int i = 0;i < 3;i++) {
                    if (i == bn) {
                        continue;
                    }
                    for (int j = 0;j < 3;j++) {
                        if (b[bn].bmem[i][j].rank < clow && b[bn].bmem[i][j].position == j && b[bn].bmem[i][j].cond>0) {
                            plow = i;
                            clow = b[bn].bmem[i][j].rank;
                            found = 1;
                            cLown = j;
                        }
                    }
                }
                if (found == 0) {
                    for (int i = 0;i < 3;i++) {
                        if (b[bn].pmem[i].rank < clow && b[bn].pmem[i].position == 3 && b[bn].pmem[i].cond>0) {
                            plow = 3;
                            clow = b[bn].pmem[i].rank;
                            found = 1;
                            cLown = i;
                        }
                    }
                    for (int i = 0;i < 3;i++) {
                        if (i == bn) {
                            continue;
                        }
                        for (int j = 0;j < 3;j++) {
                            if (b[bn].bmem[i][j].rank < clow && b[bn].bmem[i][j].position == 3 && b[bn].bmem[i][j].cond>0) {
                                plow = i;
                                clow = b[bn].bmem[i][j].rank;
                                cLown = j;
                                found = 1;
                            }
                        }
                    }
                }
                if (found == 0) {
                    plow = rand() % 4;
```

```cpp
                    while (plow == bn) {
                        plow = rand() % 4;
                    }
                    cLown = rand() % 4;
                }
                found = 0;
                for (int i = 0;i < 3;i++) {
                    if (b[bn].bmem[bn][i].rank > myHighR && b[bn].bmem[bn][i].cond > 0 && b[bn].bmem[bn][i].position == i) {
                        myHighR = b[bn].bmem[bn][i].rank;
                        myHighC = i;
                        found = 1;
                    }
                }
                if (found == 0) {
                    for (int i = 0;i < 3;i++) {
                        if (b[bn].bmem[bn][i].rank > myHighR && b[bn].bmem[bn][i].cond > 0 && b[bn].bmem[bn][i].position == 3) {
                            myHighR = b[bn].bmem[bn][i].rank;
                            myHighC = i;
                            found = 1;
                        }
                    }
                }
                if (found == 0) {
                    myHighC = rand() % 4;
                }
                int tempr, tempmr,tempmc,tempmp;
                string temps;
                switch (plow) {
                case 3:
                {
                    tempr = p.hand[cLown].rank;
                    temps = p.hand[cLown].suit;
                    p.hand[cLown].rank=b[bn].hand[myHighC].rank;
                    p.hand[cLown].suit = b[bn].hand[myHighC].suit;
                    b[bn].hand[myHighC].rank = tempr;
                    b[bn].hand[myHighC].suit = temps;
                    for (int i = 0;i < 3;i++) {
                        tempmr = b[i].pmem[cLown].rank;
                        tempmc = b[i].pmem[cLown].cond;
                        tempmp = b[i].pmem[cLown].position;
                        b[i].pmem[cLown].rank = b[i].bmem[bn][myHighC].rank;
                        b[i].pmem[cLown].cond = b[i].bmem[bn][myHighC].position;
                        b[i].pmem[cLown].position = b[i].bmem[bn][myHighC].position;
                        b[i].bmem[bn][myHighC].rank = tempmr;
                        b[i].bmem[bn][myHighC].cond = tempmc;
                        b[i].bmem[bn][myHighC].position = tempmp;
                    }
```

```cpp
            cout << "Bot exchnaged card with you" << endl;
            cout << "Your index: " << cLown << endl;
            cout << "Bot's index: " << myHighC << endl;
            break;
        }
        default:
        {
            tempr = b[plow].hand[cLown].rank;
            temps = b[plow].hand[cLown].suit;
            b[plow].hand[cLown].rank = b[bn].hand[myHighC].rank;
            b[plow].hand[cLown].suit = b[bn].hand[myHighC].suit;
            b[bn].hand[myHighC].rank = tempr;
            b[bn].hand[myHighC].suit = temps;
            for (int i = 0;i < 3;i++) {
                tempmr = b[i].bmem[plow][cLown].rank;
                tempmc = b[i].bmem[plow][cLown].cond;
                tempmp = b[i].bmem[plow][cLown].position;
                b[i].bmem[plow][cLown].rank = b[i].bmem[bn][myHighC].rank;
                b[i].bmem[plow][cLown].cond = b[i].bmem[bn][myHighC].position;
                b[i].bmem[plow][cLown].position = b[i].bmem[bn][myHighC].position;
                b[i].bmem[bn][myHighC].rank = tempmr;
                b[i].bmem[bn][myHighC].cond = tempmc;
                b[i].bmem[bn][myHighC].position = tempmp;
            }
            cout << "Bot exchnaged card with bot "<<plow << endl;
            cout << "Bot "<<plow<< " index: " << cLown << endl;
            cout << "Bot's index: " << myHighC << endl;
            break;
        }
        }
        break;
    }
    }
}

void winner() {
    int sum[4];
    for (int i = 0;i < 3;i++) {
        sum[i] = 0;
        for (int j = 0;j < 3;j++) {
            sum[i] += b[i].hand[j].rank;
        }
    }
    sum[3] = 0;
    for (int i = 0;i < 3;i++) {
        sum[3] += p.hand[i].rank;
    }
    cout << endl;
    cout << "Your score: " << sum[3]<<endl;
```

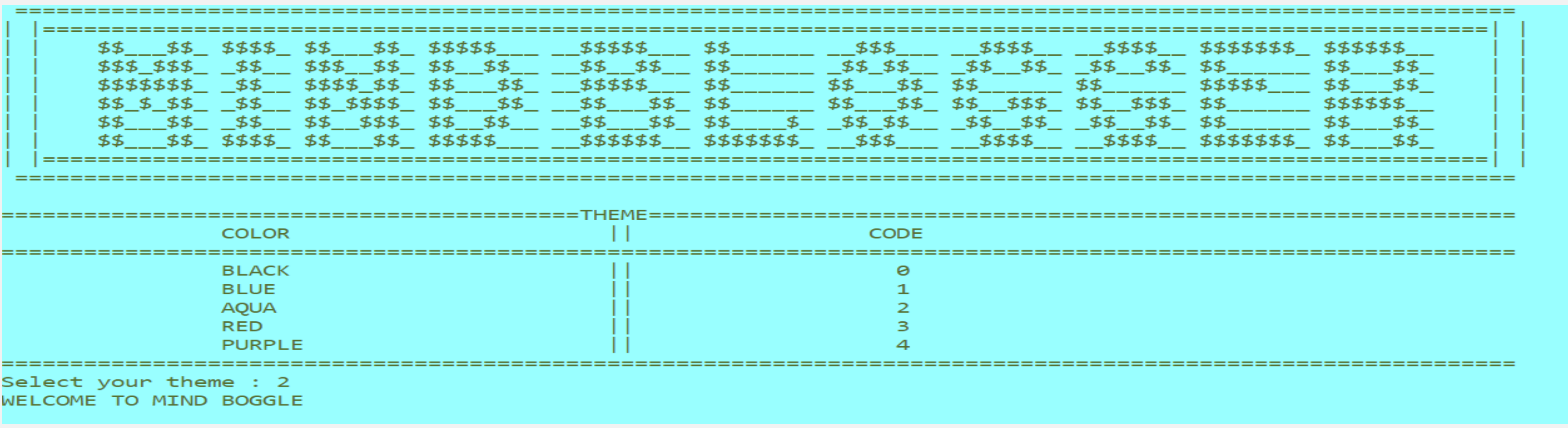
```cpp
for (int i = 0;i < 3;i++) {
    cout << "Bot " << i + 1 << " score: " << sum[i] << endl;
}
cout << endl;
if (sum[0] < sum[1] && sum[0] < sum[2] && sum[0] < sum[3]) {    //0
    cout << "BOT 1 WINS" << endl;
}
else if (sum[1] < sum[0] && sum[1] < sum[2] && sum[1] < sum[3]) {   //1
    cout << "BOT 2 WINS" << endl;
}
else if (sum[2] < sum[0] && sum[2] < sum[3] && sum[2] < sum[1]) {   //2
    cout << "BOT 3 WINS" << endl;
}
else if (sum[3] < sum[0] && sum[3] < sum[1] && sum[3] < sum[2]) {   //3
    cout << "YOU WIN" << endl;
}
else if (sum[0] == sum[1] && sum[0] < sum[2] && sum[0] < sum[3]) {  //01
    cout << "DRAW BETWEEN BOT 1 AND BOT 2" << endl;
}
else if (sum[0] == sum[2] && sum[0] < sum[3] && sum[0] < sum[1]) {  //02
    cout << "DRAW BETWEEN BOT 1 AND BOT 3" << endl;
}
else if (sum[0] == sum[3] && sum[0] < sum[2] && sum[0] < sum[1]) {  //03
    cout << "DRAW BETWEEN YOU AND BOT 1" << endl;
}
else if (sum[1] == sum[2] && sum[1] < sum[3] && sum[1] < sum[0]) {    //12
    cout << "DRAW BETWEEN BOT 2 AND BOT 3" << endl;
}
else if (sum[1] == sum[3] && sum[1] < sum[0] && sum[1] < sum[2]) {    //13
    cout << "DRAW BETWEEN YOU AND BOT 2" << endl;
}
else if (sum[2] == sum[3] && sum[2] < sum[0] && sum[2] < sum[1]) {    //23
    cout << "DRAW BETWEEN YOU AND BOT 3" << endl;
}
else if (sum[0] == sum[1] && sum[1] == sum[2] && sum[2] < sum[3]) {    //012
    cout << "DRAW BETWEEN BOT 1, BOT 2 AND BOT 3" << endl;
}
else if (sum[0] == sum[1] && sum[1] == sum[3] && sum[1] < sum[2]) {    //013
    cout << "DRAW BETWEEN YOU, BOT 1 AND BOT 2" << endl;
}
```

```cpp
        else if (sum[0] == sum[2] && sum[2] == sum[3] && sum[2] < sum[1]) {     //023
            cout << "DRAW BETWEEN YOU, BOT 1 AND BOT 3" << endl;
        }
        else if (sum[1] == sum[2] && sum[2] == sum[3] && sum[1] < sum[0]) {     //123
            cout << "DRAW BETWEEN YOU, BOT 2 AND BOT 3" << endl;
        }
        else if (sum[1] == sum[2] && sum[2] == sum[3] && sum[3] == sum[0]) {     //123
            cout << "DRAW BETWEEN ALL" << endl;
        }

    }


};

int main()
{
    srand(time(0));
    cout << "WELCOME TO MIND BOGGLE" << endl << endl;
    cout<<"RULES"<<endl<<"1) 3 cards will be distributed between 4 players, the player will look at those cards once and put them upside down.\n2) Each player can pick one card from deck.Now he can either discard that card or exchange it with one of his three cards(he cannot look his 3 cards again).There would be 40 cards in deck.\n3)  If a player discards card ranked 10, he can pick any opponent’s cards and shuffle them(now opponent doesn’t know which card is where.\n4) If a player discards card ranked 7, he can exchange one of his card with any of opponent’s card without looking at both cards."<<endl;
    cout << "5) If a player discards card ranked ‘King’, he can pick his 3 cards, look at them and shuffle them.\n6) The game will end after all cards from deck are picked up.\n7) Player whose sum of card rank is lowest will win." << endl;
    Game g;
    int x = 0;
    g.initializeDeck();
    g.distributeCards();
    cout << "\nYour cards are:" << endl;
    g.p.printHand();
    for (int i = 0;i < 10;i++) {
        cout << endl;
        g.pmove(g.bank[x].rank, g.bank[x].suit);
        x++;
        for (int j = 0;j < 3;j++) {
            cout << endl;
            g.bmove(g.bank[x].rank, g.bank[x].suit, j);
            x++;
        }
    }
    g.printAllHands();
```

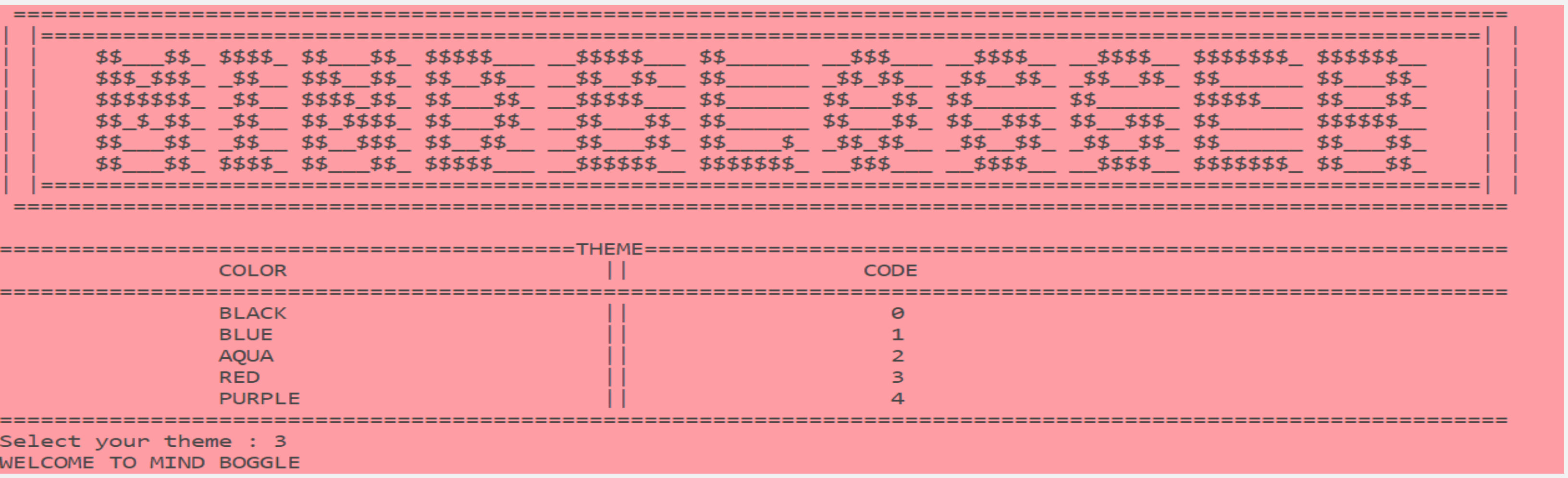
```
        g.winner();

    }
```

10. OUTPUTS:

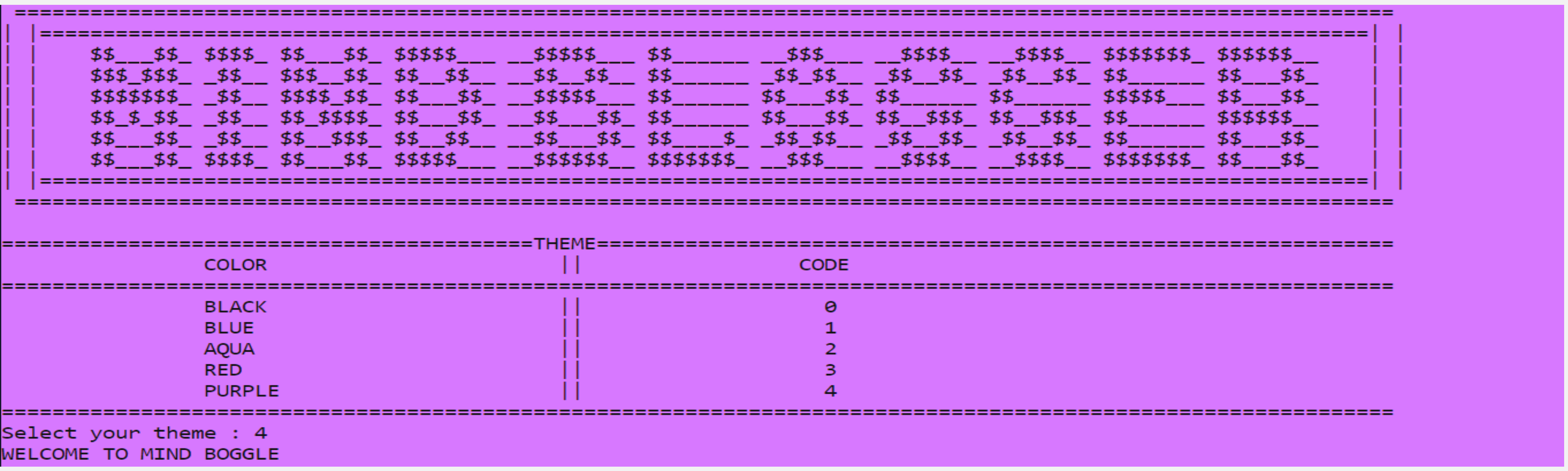
DIFFERENT THEMES



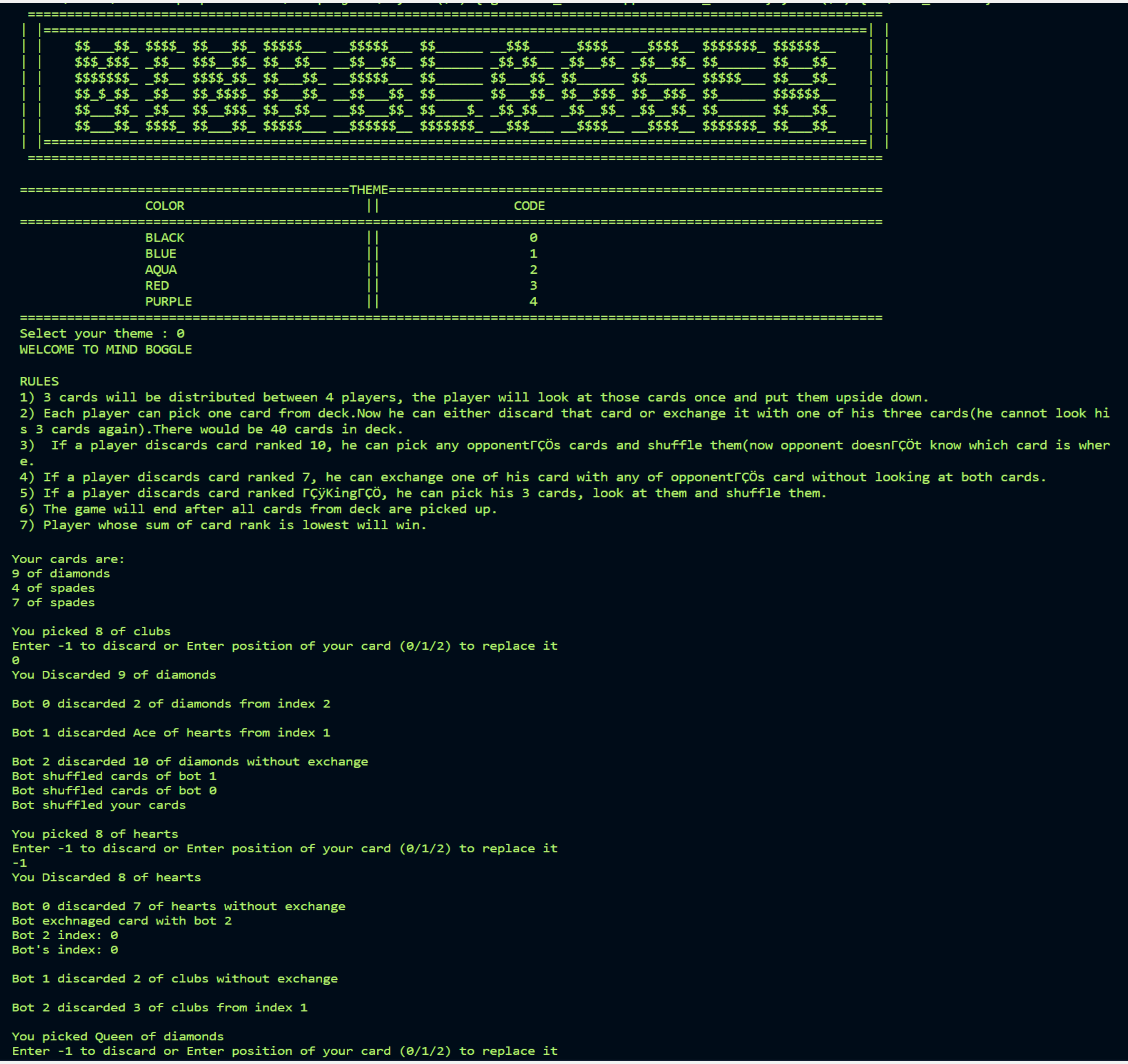
```
 ==============================================================================================================
| |==========================================================================================================| |
| |     $$___$$_ $$$$_ $$___$$_ $$$$$___ __$$$$$___ $$______ __$$$___ __$$$$__ __$$$$__ $$$$$$$_ $$$$$$__      | |
| |     $$$_$$$_ _$$__ $$$__$$_ $$__$$__ __$$__$$__ $$______ _$$_$$__ _$$__$$_ _$$__$$_ $$______ $$___$$_      | |
| |     $$$$$$$_ _$$__ $$$$_$$_ $$___$$_ __$$$$$___ $$______ $$___$$_ $$______ $$______ $$$$$___ $$___$$_      | |
| |     $$_$_$$_ _$$__ $$_$$$$_ $$___$$_ __$$___$$_ $$______ $$___$$_ $$__$$$_ $$__$$$_ $$______ $$$$$$__      | |
| |     $$___$$_ _$$__ $$__$$$_ $$__$$__ __$$___$$_ $$____$_ _$$_$$__ _$$__$$_ _$$__$$_ $$______ $$___$$_      | |
| |     $$___$$_ $$$$_ $$___$$_ $$$$$___ __$$$$$$__ $$$$$$$_ __$$$___ __$$$$__ __$$$$__ $$$$$$$_ $$___$$_      | |
| |==========================================================================================================| |
 ==============================================================================================================

==========================================THEME=================================================================
                COLOR                        ||                  CODE
================================================================================================================
                BLACK                        ||                   0
                BLUE                         ||                   1
                AQUA                         ||                   2
                RED                          ||                   3
                PURPLE                       ||                   4
================================================================================================================
Select your theme : 2
WELCOME TO MIND BOGGLE
```



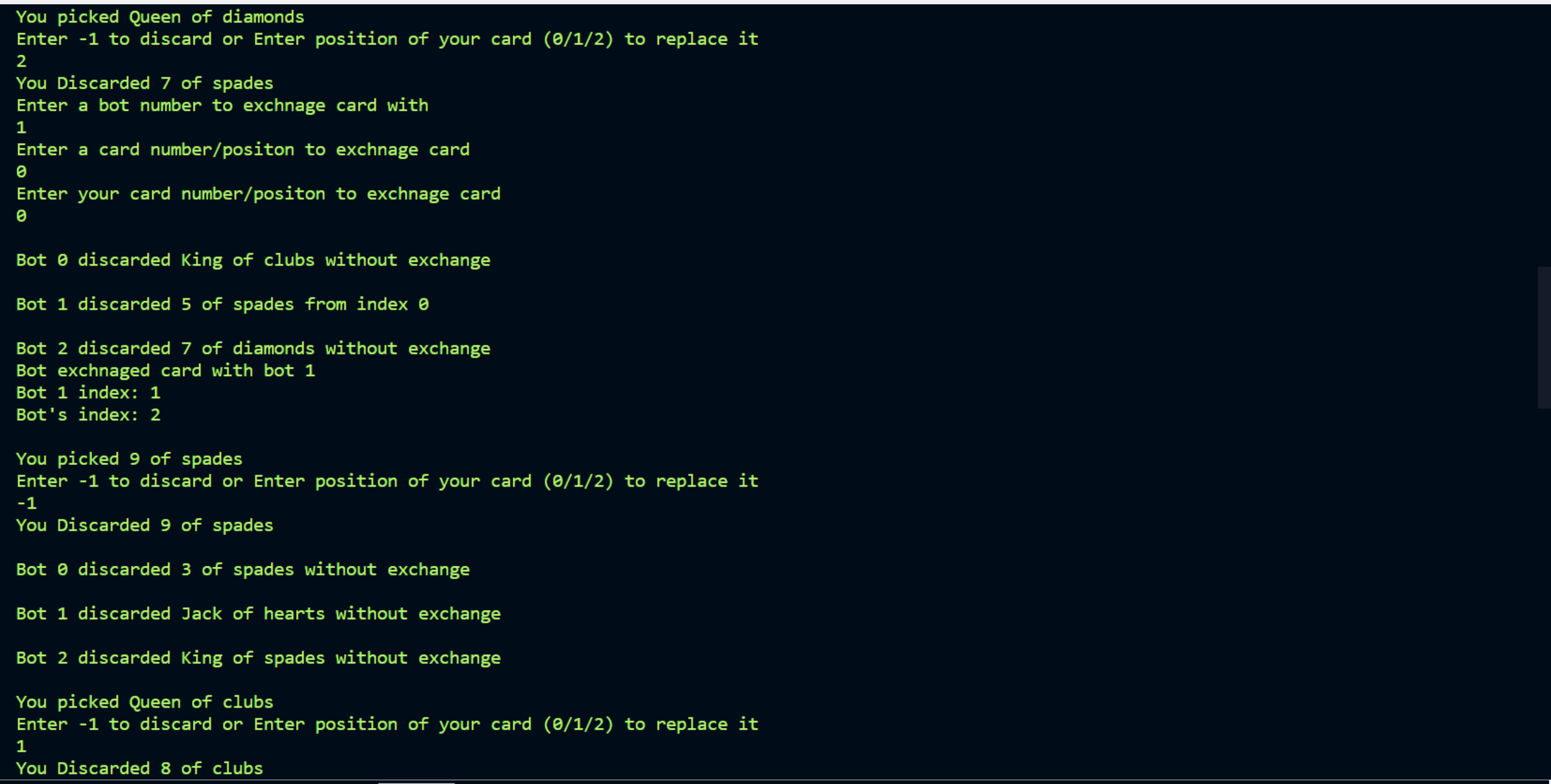
```
 ==============================================================================================================
| |==========================================================================================================| |
| |     $$___$$_ $$$$_ $$___$$_ $$$$$___ __$$$$$___ $$______ __$$$___ __$$$$__ __$$$$__ $$$$$$$_ $$$$$$__      | |
| |     $$$_$$$_ _$$__ $$$__$$_ $$__$$__ __$$__$$__ $$______ _$$_$$__ _$$__$$_ _$$__$$_ $$______ $$___$$_      | |
| |     $$$$$$$_ _$$__ $$$$_$$_ $$___$$_ __$$$$$___ $$______ $$___$$_ $$______ $$______ $$$$$___ $$___$$_      | |
| |     $$_$_$$_ _$$__ $$_$$$$_ $$___$$_ __$$___$$_ $$______ $$___$$_ $$__$$$_ $$__$$$_ $$______ $$$$$$__      | |
| |     $$___$$_ _$$__ $$__$$$_ $$__$$__ __$$___$$_ $$____$_ _$$_$$__ _$$__$$_ _$$__$$_ $$______ $$___$$_      | |
| |     $$___$$_ $$$$_ $$___$$_ $$$$$___ __$$$$$$__ $$$$$$$_ __$$$___ __$$$$__ __$$$$__ $$$$$$$_ $$___$$_      | |
| |==========================================================================================================| |
 ==============================================================================================================

==========================================THEME=================================================================
                COLOR                        ||                  CODE
================================================================================================================
                BLACK                        ||                   0
                BLUE                         ||                   1
                AQUA                         ||                   2
                RED                          ||                   3
                PURPLE                       ||                   4
================================================================================================================
Select your theme : 3
WELCOME TO MIND BOGGLE
```



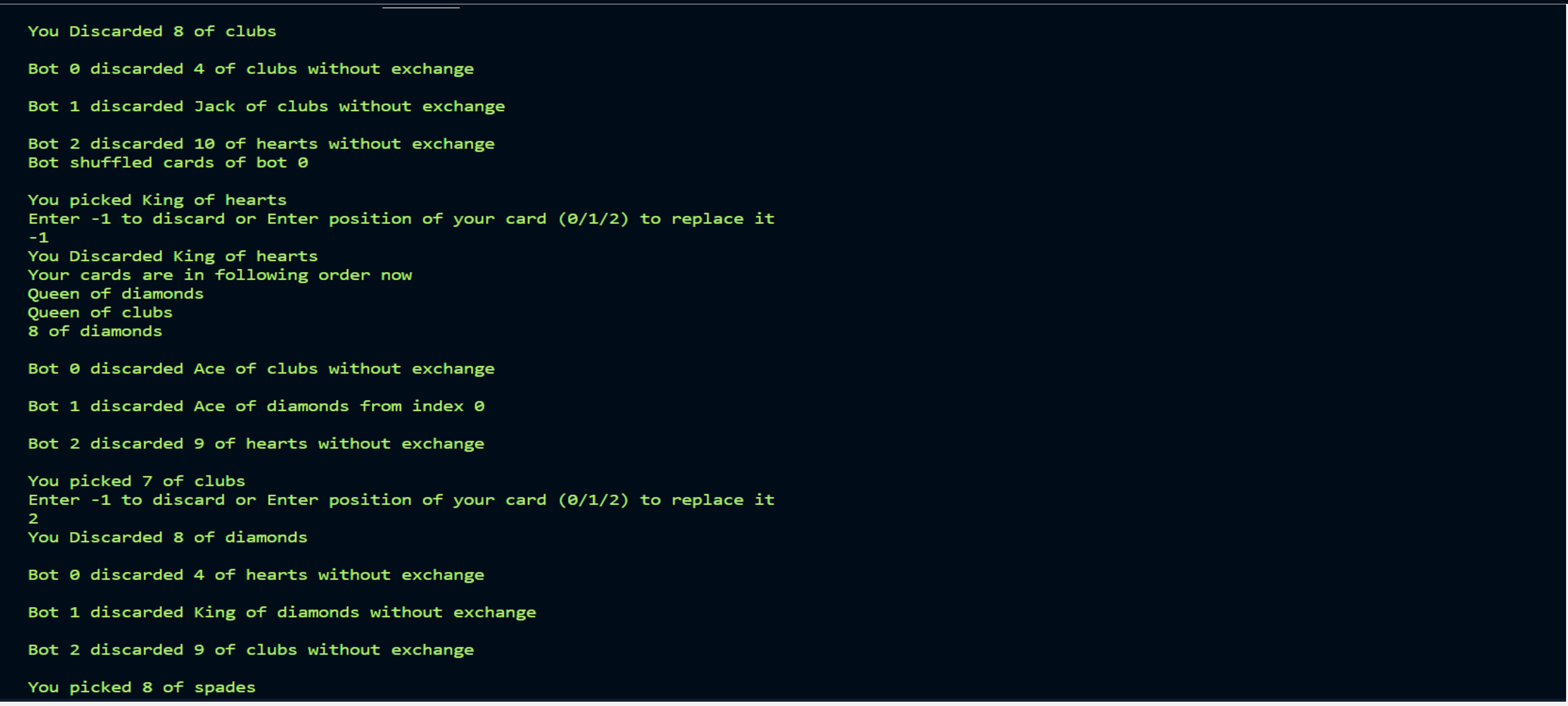
```
 ==============================================================================================================
| |==========================================================================================================| |
| |     $$___$$_ $$$$_ $$___$$_ $$$$$___ __$$$$$___ $$______ __$$$___ __$$$$__ __$$$$__ $$$$$$$_ $$$$$$__      | |
| |     $$$_$$$_ _$$__ $$$__$$_ $$__$$__ __$$__$$__ $$______ _$$_$$__ _$$__$$_ _$$__$$_ $$______ $$___$$_      | |
| |     $$$$$$$_ _$$__ $$$$_$$_ $$___$$_ __$$$$$___ $$______ $$___$$_ $$______ $$______ $$$$$___ $$___$$_      | |
| |     $$_$_$$_ _$$__ $$_$$$$_ $$___$$_ __$$___$$_ $$______ $$___$$_ $$__$$$_ $$__$$$_ $$______ $$$$$$__      | |
| |     $$___$$_ _$$__ $$__$$$_ $$__$$__ __$$___$$_ $$____$_ _$$_$$__ _$$__$$_ _$$__$$_ $$______ $$___$$_      | |
| |     $$___$$_ $$$$_ $$___$$_ $$$$$___ __$$$$$$__ $$$$$$$_ __$$$___ __$$$$__ __$$$$__ $$$$$$$_ $$___$$_      | |
| |==========================================================================================================| |
 ==============================================================================================================

==========================================THEME=================================================================
                COLOR                        ||                  CODE
================================================================================================================
                BLACK                        ||                   0
                BLUE                         ||                   1
                AQUA                         ||                   2
                RED                          ||                   3
                PURPLE                       ||                   4
================================================================================================================
Select your theme : 4
WELCOME TO MIND BOGGLE
```



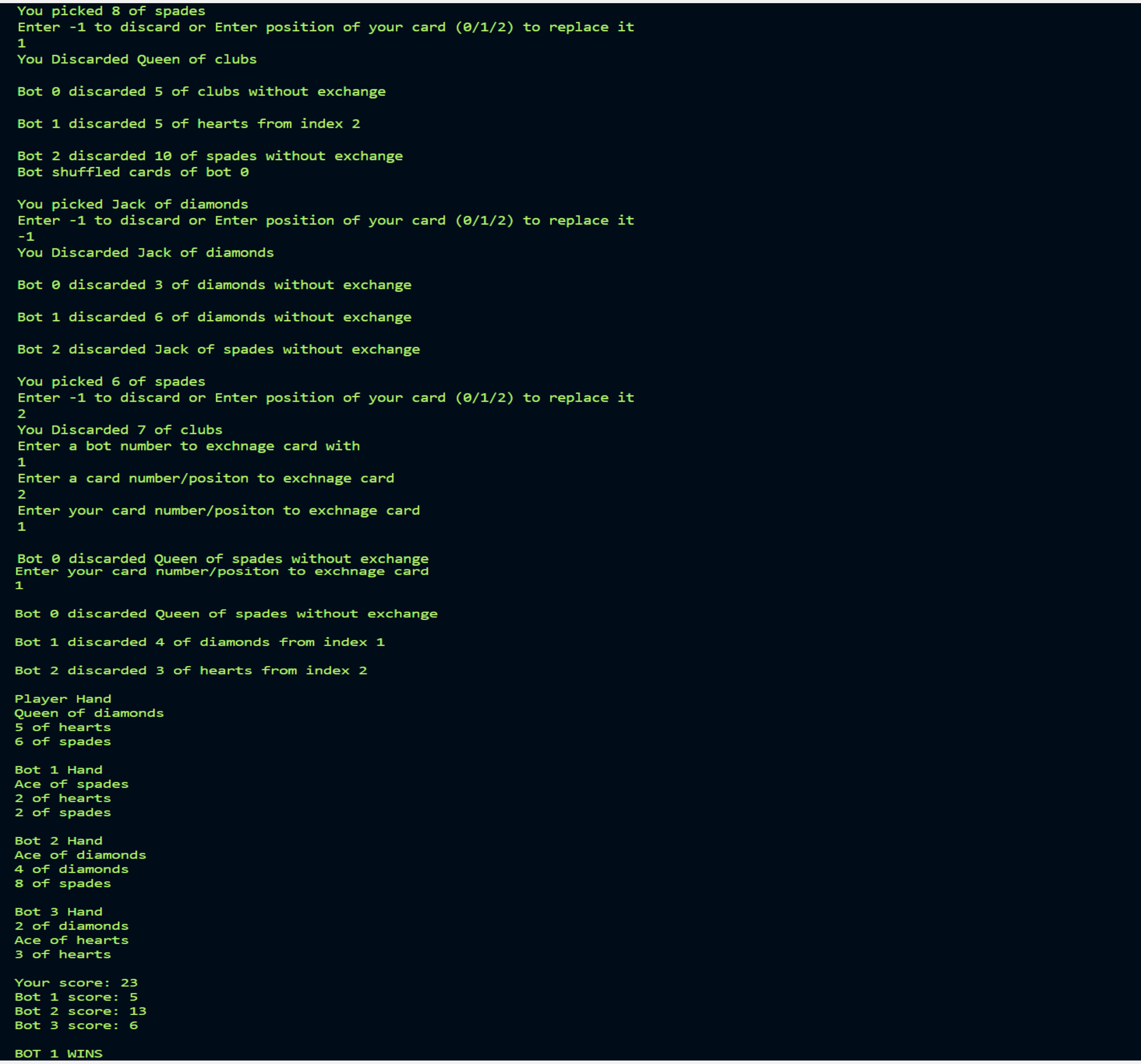
```
 ===============================================================================================================
| |=============================================================================================================| |
| |    $$___$$_ $$$$_ $$___$$_ $$$$$___ __$$$$$___ $$______ __$$$___ __$$$$__ __$$$$__ $$$$$$$_ $$$$$$__    | |
| |    $$$_$$$_ _$$__ $$$__$$_ $$__$$__ __$$__$$__ $$______ _$$_$$__ _$$__$$_ _$$__$$_ $$______ $$___$$_    | |
| |    $$$$$$$_ _$$__ $$$$_$$_ $$___$$_ __$$$$$___ $$______ $$___$$_ $$______ $$______ $$$$$___ $$___$$_    | |
| |    $$_$_$$_ _$$__ $$_$$$$_ $$___$$_ __$$___$$_ $$______ $$___$$_ $$__$$$_ $$__$$$_ $$______ $$$$$$__    | |
| |    $$___$$_ _$$__ $$__$$$_ $$__$$__ __$$___$$_ $$____$_ _$$_$$__ _$$__$$_ _$$__$$_ $$______ $$___$$_    | |
| |    $$___$$_ $$$$_ $$___$$_ $$$$$___ __$$$$$$__ $$$$$$$_ __$$$___ __$$$$__ __$$$$__ $$$$$$$_ $$___$$_    | |
| |=============================================================================================================| |
 ===============================================================================================================

==========================================THEME=================================================================
                COLOR                           ||                  CODE
================================================================================================================
                BLACK                           ||                  0
                BLUE                            ||                  1
                AQUA                            ||                  2
                RED                             ||                  3
                PURPLE                          ||                  4
================================================================================================================
Select your theme : 0
WELCOME TO MIND BOGGLE

RULES
1) 3 cards will be distributed between 4 players, the player will look at those cards once and put them upside down.
2) Each player can pick one card from deck.Now he can either discard that card or exchange it with one of his three cards(he cannot look hi
s 3 cards again).There would be 40 cards in deck.
3)  If a player discards card ranked 10, he can pick any opponentΓÇÖs cards and shuffle them(now opponent doesnΓÇÖt know which card is wher
e.
4) If a player discards card ranked 7, he can exchange one of his card with any of opponentΓÇÖs card without looking at both cards.
5) If a player discards card ranked ΓÇÿKingΓÇÖ, he can pick his 3 cards, look at them and shuffle them.
6) The game will end after all cards from deck are picked up.
7) Player whose sum of card rank is lowest will win.

Your cards are:
9 of diamonds
4 of spades
7 of spades

You picked 8 of clubs
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
0
You Discarded 9 of diamonds

Bot 0 discarded 2 of diamonds from index 2

Bot 1 discarded Ace of hearts from index 1

Bot 2 discarded 10 of diamonds without exchange
Bot shuffled cards of bot 1
Bot shuffled cards of bot 0
Bot shuffled your cards

You picked 8 of hearts
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
-1
You Discarded 8 of hearts

Bot 0 discarded 7 of hearts without exchange
Bot exchnaged card with bot 2
Bot 2 index: 0
Bot's index: 0

Bot 1 discarded 2 of clubs without exchange

Bot 2 discarded 3 of clubs from index 1

You picked Queen of diamonds
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
```



```
You picked Queen of diamonds
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
2
You Discarded 7 of spades
Enter a bot number to exchnage card with
1
Enter a card number/positon to exchnage card
0
Enter your card number/positon to exchnage card
0

Bot 0 discarded King of clubs without exchange

Bot 1 discarded 5 of spades from index 0

Bot 2 discarded 7 of diamonds without exchange
Bot exchnaged card with bot 1
Bot 1 index: 1
Bot's index: 2

You picked 9 of spades
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
-1
You Discarded 9 of spades

Bot 0 discarded 3 of spades without exchange

Bot 1 discarded Jack of hearts without exchange

Bot 2 discarded King of spades without exchange

You picked Queen of clubs
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
1
You Discarded 8 of clubs
```



```
You Discarded 8 of clubs

Bot 0 discarded 4 of clubs without exchange

Bot 1 discarded Jack of clubs without exchange

Bot 2 discarded 10 of hearts without exchange
Bot shuffled cards of bot 0

You picked King of hearts
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
-1
You Discarded King of hearts
Your cards are in following order now
Queen of diamonds
Queen of clubs
8 of diamonds

Bot 0 discarded Ace of clubs without exchange

Bot 1 discarded Ace of diamonds from index 0

Bot 2 discarded 9 of hearts without exchange

You picked 7 of clubs
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
2
You Discarded 8 of diamonds

Bot 0 discarded 4 of hearts without exchange

Bot 1 discarded King of diamonds without exchange

Bot 2 discarded 9 of clubs without exchange

You picked 8 of spades
```



```
You picked 8 of spades
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
1
You Discarded Queen of clubs

Bot 0 discarded 5 of clubs without exchange

Bot 1 discarded 5 of hearts from index 2

Bot 2 discarded 10 of spades without exchange
Bot shuffled cards of bot 0

You picked Jack of diamonds
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
-1
You Discarded Jack of diamonds

Bot 0 discarded 3 of diamonds without exchange

Bot 1 discarded 6 of diamonds without exchange

Bot 2 discarded Jack of spades without exchange

You picked 6 of spades
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
2
You Discarded 7 of clubs
Enter a bot number to exchnage card with
1
Enter a card number/positon to exchnage card
2
Enter your card number/positon to exchnage card
1

Bot 0 discarded Queen of spades without exchange
Enter your card number/positon to exchnage card
1

Bot 0 discarded Queen of spades without exchange

Bot 1 discarded 4 of diamonds from index 1

Bot 2 discarded 3 of hearts from index 2

Player Hand
Queen of diamonds
5 of hearts
6 of spades

Bot 1 Hand
Ace of spades
2 of hearts
2 of spades

Bot 2 Hand
Ace of diamonds
4 of diamonds
8 of spades

Bot 3 Hand
2 of diamonds
Ace of hearts
3 of hearts

Your score: 23
Bot 1 score: 5
Bot 2 score: 13
Bot 3 score: 6

BOT 1 WINS
```

11.HOW MUCH DID WE SUCCEED :

We was able to make the game which was interesting and inruiting , however we wasn't able to implement some algorithms which we think can be much much attractive for user and the mechanism we expected to use with cards was also different as we expect because of difficulty we face while try adding card ascii cart still we was able to make a game which can boggle your mind and make you challenge yourself to win which is the most important thing for developers to make sure that user hooks with the project he created.

THE END!