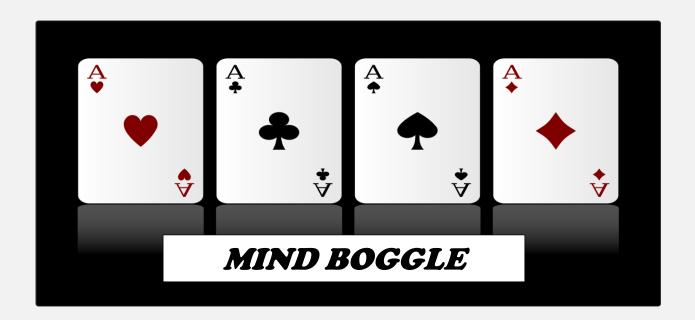
OBJECT ORIENTED PROGRAMMING [CT-260]

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.

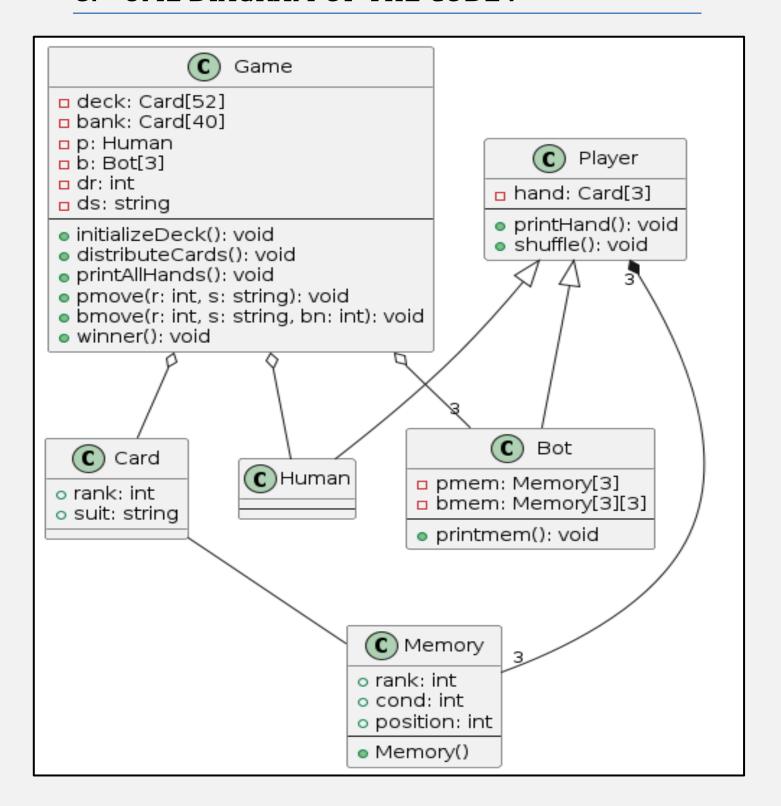
RANDOM NUMBER GENERATION: The code utilizes the rand() function to generate random numbers for shuffling cards, distributing cards, and making decisions during the game. This demonstrates the use of random number generation in a program.

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:

```
1. #include <iostream>
2. #include <string>
3. #include <ctime>
4. #include <cstdlib>
5. using namespace std;
6. class Card {
7. public:
8. int rank;
9.
     string suit;
10. };
11.
12. class Memory (
13.
      public:
14.
       int rank, cond, position;
                                                       // store rank,
  condition (known, less than, unknown) and position of own and opponents
15.
          Memory() {
16.
              rank = 14;
                                               //default rank = 14 means
 unknown
17.
                                               // default condition = 0 means
               cond = 0;
unknown
18.
19. }
              position = 3;
                                              //3 means unknown
19.
20. };
21.
22. class Player {
23. public:
        Card hand[3];
24.
25.
26.
       void printHand() {
                                                   //prints card on hand
27.
               for (int i = 0;i < 3;i++) {</pre>
28.
                   switch (hand[i].rank) {
29.
                   case 1:
30.
                       cout << "Ace";
31.
                      break;
32.
                    case 11:
33.
                       cout << "Jack";</pre>
34.
                       break;
35.
                   case 12:
36.
                       cout << "Queen";</pre>
37.
                       break;
38.
                    case 13:
39.
                       cout << "King";</pre>
40.
                       break;
41.
                   default:
42.
                      cout << hand[i].rank;</pre>
43.
                   cout << " of " << hand[i].suit << endl;</pre>
44.
45.
```

```
46.
                                                           //shuffles card of a
47.
            void shuffle() {
 hand
48.
                int y = rand() % 3;
49.
                int z = rand() % 3;
50.
                int tempr;
51.
                string temps;
52.
                for (int i = 0;i < 10;i++) {</pre>
53.
                     tempr = hand[y].rank;
54.
                     temps = hand[y].suit;
55.
                     hand[y].rank = hand[z].rank;
56.
                    hand[y].suit = hand[z].suit;
57.
                    hand[z].rank = tempr;
58.
                    hand[z].suit = temps;
59.
                    y = rand() % 3;
60.
                    z = rand() % 3;
61.
                }
62.
            }
63.
        };
64.
65.
        class Human :public Player {
66.
        };
67.
        class Bot:public Player {
68.
69.
        public:
70.
            Memory pmem[3], bmem[3][3];
71.
72.
           void printmem() {
                                                                //prints memory of
 a bot... func for test only
73.
                cout << "Player cards: " << endl;</pre>
74.
                for (int i = 0;i < 3;i++) {</pre>
                    cout << "Position: " << pmem[i].position << endl << "Rank:</pre>
" << pmem[i].rank << endl << "Condition: " << pmem[i].cond << endl;
76.
77.
                cout << "Bot Cards:" << endl << endl ;</pre>
78.
                for (int i = 0;i < 3;i++) {</pre>
                     cout << "Bot " << i << endl;
79.
80.
                     for (int j = 0; j < 3; j++) {
                         cout << "Position: " << bmem[i][j].position << endl <<</pre>
  "Rank: " << bmem[i][j].rank << endl << "Condition: " << bmem[i][j].cond <<
  endl << endl;</pre>
82.
83.
                }
84.
85.
86.
       };
87.
88.
        class Game {
89.
        public:
90.
            Card deck[52];
91.
           Card bank[40];
            Human p;
92.
```

```
93.
         Bot b[3];
94.
         int dr;
                                 //discarded rank
95.
                                  //discarded suit
         string ds;
         void initializeDeck() {
                                         //intializes deck in
sorted order
97.
             int k = 0;
              string s[4] = {"spades", "clubs", "hearts", "diamonds"};
98.
99.
             for (int i = 0;i < 4;i++) {</pre>
100.
               for (int j = 1; j < 14; j++) {
101.
                     deck[k].rank = j;
102.
                     deck[k].suit = s[i];
103.
                    k++;
104.
105.
             }
106.
         }
107.
         void distributeCards() {
108.
109.
110.
             int y = rand() % 52;
              for (int i = 0;i < 3;i++) {
111.
                                                //dist card to player
112.
                 while (deck[y].rank == 0) {
                   y = rand() % 52;
113.
                                                 //random num to get
card randomly from the deck
114.
                }
115.
                p.hand[i].rank = deck[y].rank;
                p.hand[i].suit = deck[y].suit;
116.
117.
                deck[y].rank = 0;
                                                  //setting value for
the rank to 0, since this card is already distributed...
118.
119.
             for (int i = 0;i < 3;i++) {</pre>
                                                 //dist card to bots...
i is for bot whos recieving card
120.
                for (int j = 0; j < 3; j++) { // j is for position
of card
121.
                     while (deck[y].rank == 0) {
122.
                     y = rand() % 52;
123.
124.
                     b[i].hand[j].rank = deck[y].rank;
125.
                     b[i].hand[j].suit = deck[y].suit;
                     b[i].bmem[i][j].rank = deck[y].rank;  // bot
 memorizes its cards here
127.
                    b[i].bmem[i][j].position = j; //3 means
 unknown
128.
                    b[i].bmem[i][j].cond = 2;
                                                        // 0 unknown,
1 less than=, 2 known
129.
                     deck[y].rank = 0;
130.
                 }
131.
             }
132.
            for (int i = 0;i < 40;i++) {</pre>
                                                //distributing
remaining cards to bank randomly
134.
          while (deck[y].rank == 0) {
135.
                 y = rand() % 52;
```

```
136.
137.
                     bank[i].rank = deck[y].rank;
138.
                     bank[i].suit = deck[y].suit;
139.
                     deck[y].rank = 0;
140.
                }
141.
            }
142.
143.
            void printAllHands() {
                                                                     //prints all
  hands in game... use in end
144.
                cout << endl << "Player Hand" << endl;</pre>
145.
                 p.printHand();
146.
                 for (int i = 0; i < 3; i++) {
                     cout << endl << "Bot " << i + 1 << " Hand" << endl;</pre>
147.
148.
                    b[i].printHand();
149.
                }
150.
           }
151.
152.
            void pmove(int r, string s) {
                                                                //players move
153.
                int m;
                 cout << "You picked ";</pre>
154.
155.
                 switch (r)
156.
157.
                 case 1:
                     cout << "Ace";
158.
159.
                     break;
160.
                 case 11:
161.
                    cout << "Jack";</pre>
162.
                    break;
163.
                 case 12:
                    cout << "Queen";</pre>
164.
165.
                    break;
166.
                 case 13:
167.
                     cout << "King";</pre>
168.
                     break;
169.
                default:
170.
                    cout << r;
171.
                 }
                 cout << " of " << s << endl;
172.
                 cout << "Enter -1 to discard or Enter position of your card</pre>
(0/1/2) to replace it" << endl;</pre>
174.
                cin >> m;
175.
                 while (m > 2 \mid | m < -1) {
176.
                     cout << "Invalid number, enter again" << endl;</pre>
177.
178.
                 switch (m) {
179.
                 case -1:
180.
                    dr = r;
181.
                     ds = s;
182.
                     for (int i = 0;i < 3;i++) {</pre>
183.
                         for (int j = 0; j < 3; j++) {
184.
                              if (b[i].pmem[j].rank > dr) {
185.
                                  b[i].pmem[j].rank = dr;
```

```
186.
                                  b[i].pmem[j].cond = 1;
187.
                              }
188.
                         }
189.
                     }
190.
                     break;
191.
                 default:
192.
                     dr = p.hand[m].rank;
193.
                     ds = p.hand[m].suit;
194.
                     p.hand[m].rank = r;
195.
                     p.hand[m].suit = s;
196.
197.
                     for (int i = 0; i < 3; i++) {
198.
                         b[i].pmem[m].cond = 1;
199.
                         b[i].pmem[m].position = m;
200.
                         b[i].pmem[m].rank = dr;
201.
                    }
202.
                 }
203.
                 cout << "You Discarded ";</pre>
204.
                 switch (dr) {
205.
                 case 1:
206.
                     cout << "Ace";</pre>
207.
                     break;
208.
                 case 11:
                     cout << "Jack";</pre>
209.
210.
                     break;
211.
                 case 12:
212.
                     cout << "Queen";</pre>
213.
                     break;
214.
                 case 13:
215.
                     cout << "King";</pre>
216.
                     break;
217.
                 default:
                     cout << dr;
218.
219.
220.
                 cout << " of " << ds << endl;
221.
                 switch (dr) {
222.
                 case 13:
                                           //King thrown.. see your card
223.
                 {
224.
                     p.shuffle();
225.
                     cout << "Your cards are in following order now" << endl;</pre>
226.
                     p.printHand();
227.
                     for (int i = 0;i < 3;i++) {</pre>
228.
                         for (int j = 0; j < 3; j++) {
229.
                             b[i].pmem[j].position = 3;
230.
231.
                     }
232.
                     break;
233.
                 }
234.
                 case 10:
                                     //10 ...shuffle oppponent's card
235.
                 {
236.
                     int x;
```

```
cout << "Enter a number to shuffle bot's cards and view</pre>
237.
  them" << endl;
238.
                    cin >> x;
239.
                    while (x < 0 | | x>2) {
240.
                        cout << "Invalid input, enter the bot number again" <<</pre>
  endl;
241.
                        cin >> x;
242.
                    }
243.
                    b[x].shuffle();
244.
                    cout << "Bot " << x << "'s cards are in this order now" <<
  endl;
245.
                   b[x].printHand();
246.
                    for (int i = 0;i < 3;i++) {</pre>
247.
                        for (int j = 0; j < 3; j++) {
248.
                            b[i].bmem[x][j].position = 3;
249.
250.
                    }
251.
                    break;
252.
                }
253.
                case 7:
                                   //7... exchange card w/o looking
254.
255.
                    int x, y, z;
256.
                    cout << "Enter a bot number to exchnage card with" <<</pre>
 endl;
257.
                    cin >> x;
258.
                    while (x < 0 | | x>2) {
259.
                        cout << "Invalid input, enter the bot number again" <<</pre>
  endl;
260.
                        cin >> x;
261.
                   }
262.
                    cout << "Enter a card number/position to exchange card" <</pre>
  endl;
263.
                    cin >> y;
264.
                    while (y < 0 | | y>2) {
265.
                        cout << "Invalid input, enter the card number again"</pre>
 << endl;
266.
                       cin >> y;
267.
                    }
                    cout << "Enter your card number/positon to exchnage card"</pre>
  << endl;
269.
                    cin >> z;
270.
                    while (z < 0 | | z>2) {
271.
                        cout << "Invalid input, enter the card number again"</pre>
 << endl;
272.
                        cin >> z;
273.
                    int tempr, tempmr, tempmc, tempmp; //tempmr is for
274.
 memory rank...
275.
                    string temps;
276.
                    tempr = p.hand[z].rank;
277.
                    temps = p.hand[z].suit;
278.
                    p.hand[z].rank = b[x].hand[y].rank;
```

```
279.
                     p.hand[z].suit = b[x].hand[y].suit;
280.
                     b[x].hand[y].rank=tempr;
281.
                     b[x].hand[y].suit=temps;
282.
                     for (int i = 0;i < 3;i++) {</pre>
283.
                         tempmr = b[i].pmem[z].rank;
284.
                         tempmc = b[i].pmem[z].cond;
285.
                         tempmp = b[i].pmem[z].position;
286.
                         b[i].pmem[z].rank=b[i].bmem[x][y].rank;
287.
                         b[i].pmem[z].cond = b[i].bmem[x][y].position;
288.
                         b[i].pmem[z].position = b[i].bmem[x][y].position;
289.
                         b[i].bmem[x][y].rank=tempmr;
290.
                         b[i].bmem[x][y].cond = tempmc;
291.
                         b[i].bmem[x][y].position = tempmp;
292.
293.
                     break;
294.
                 }
295.
                }
296.
            }
297.
298.
            void bmove(int r, string s, int bn) {
299.
                int pos = 4;
300.
                bool discard flag = 0;
301.
                int z = rand() % 5;
302.
                int myHighR = 0;
303.
                int myHighC=4;
304.
                if (r <= 6) {
305.
                     for (int i = 0;i < 3;i++) {</pre>
306.
                         if (b[bn].bmem[bn][i].rank > myHighR &&
  b[bn].bmem[bn][i].cond > 0 && b[bn].bmem[bn][i].position == i) {
307.
                             myHighR = b[bn].bmem[bn][i].rank;
308.
                             myHighC = i;
309.
                             discard flag = 1;
310.
                         }
311.
312.
                     if (discard flag == 0) {
313.
                         if (z == 0) {
314.
                             goto 11;
315.
316.
                         for (int i = 0; i < 3; i++) {
317.
                             if (b[bn].bmem[bn][i].rank > myHighR &&
  b[bn].bmem[bn][i].cond > 0 && b[bn].bmem[bn][i].position == 3) {
318.
                                 myHighR = b[bn].bmem[bn][i].rank;
319.
                                 myHighC = i;
320.
                                 discard flag = 1;
321.
322.
323.
                     }
324.
                     11:
325.
                     if (discard flag == 1) {
326.
                         if (myHighR < r) {</pre>
327.
                             discard flag = 0;
328.
```

```
329.
330.
                     if (discard flag == 1) {
331.
                         b[bn].hand[myHighC].rank = r;
332.
                         b[bn].hand[myHighC].suit = s;
333.
                         dr = b[bn].hand[myHighC].rank;
334.
                          ds = b[bn].hand[myHighC].suit;
335.
                         b[bn].bmem[bn][myHighC].cond = 2;
336.
                         b[bn].bmem[bn][myHighC].position = myHighC;
337.
                         b[bn].bmem[bn][myHighC].rank = r;
338.
                         pos = myHighC;
339.
                         for (int i = 0;i < 3;i++) {</pre>
340.
                              if (i == bn) {
341.
                                  continue;
342.
                              }
343.
                              b[i].bmem[bn][myHighC].cond = 1;
344.
                              b[i].bmem[bn][myHighC].position = myHighC;
345.
                              b[i].bmem[bn][myHighC].rank = dr;
346.
                          }
347.
                     }
348.
349.
                 if (discard flag == 0) {
350.
                     dr = r;
351.
                     ds = s;
352.
                     for (int i = 0; i < 3; i++) {
353.
                         if (i == bn) {
354.
                              continue;
355.
356.
                          for (int j = 0; j < 3; j++) {
357.
                              if (b[i].bmem[bn][j].cond == 0) {
358.
                                  b[i].bmem[bn][j].cond = 1;
359.
                                  b[i].bmem[bn][j].position = j;
360.
                                  b[i].bmem[bn][j].rank = dr;
361.
362.
                          }
363.
                     }
364.
                 }
365.
                 cout << "Bot " << bn << " discarded ";</pre>
366.
367.
                 switch (dr) {
                 case 1:
368.
369.
                     cout << "Ace";
370.
                     break;
371.
                 case 11:
372.
                     cout << "Jack";</pre>
373.
                     break;
374.
                 case 12:
375.
                     cout << "Queen";
376.
                     break;
377.
                 case 13:
378.
                     cout << "King";</pre>
379.
                     break;
380.
                 default:
```

```
381.
                     cout << dr;
382.
                }
383.
                cout << " of " << ds;
384.
                switch (pos) {
385.
                case 4: {
386.
                    cout << " without exchange" << endl;</pre>
387.
                     break;
388.
                }
389.
                default:
390.
391.
                     cout << " from index " << pos << endl;</pre>
392.
393.
394.
                switch (dr) {
395.
                case 13:
                                              //bots move on king
396.
397.
                    b[bn].shuffle();
398.
                     for (int j = 0; j < 3; j++) {
399.
                         b[bn].bmem[bn][j].rank = b[bn].hand[j].rank;
400.
                         b[bn].bmem[bn][j].position = j;
401.
                         b[bn].bmem[bn][j].cond = 2;
402.
403.
                     for (int i = 0; i < 3; i++) {
404.
                         if (i == bn) {
405.
                             continue;
406.
407.
                         for (int j = 0; j < 3; j++) {
408.
                            b[i].bmem[bn][j].position = 3;
409.
410.
                     }
411.
                    break;
412.
413.
                case 10:
                                          //bot's move on 10
414.
415.
                     int x = 0;
                     int z = 0;
416.
417.
                    int y = rand() % 4;
418.
                    while (x < 6) {
419.
                         while (y == bn) {
420.
                             y = rand() % 4;
421.
422.
                         switch (y) {
423.
                         case 3:
424.
425.
                             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)) {
426.
                                 break;
427.
428.
                             p.shuffle();
429.
                             cout << "Bot shuffled your cards" << endl;</pre>
```

```
430.
                             z = 1:
431.
                             for (int j = 0; j < 3; j++) {
432.
                                  b[bn].pmem[j].rank = p.hand[j].rank;
433.
                                  b[bn].pmem[j].position = j;
434.
                                  b[bn].pmem[j].cond = 2;
435.
436.
                             for (int i = 0; i < 3; i++) {
437.
                                  if (i == bn) {
438.
                                      continue;
439.
440.
                                  for (int j = 0; j < 3; j++) {
441.
                                      b[i].pmem[j].position = 3;
442.
443.
444.
                             break;
445.
446.
                         default:
447.
448.
                             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)) {
449.
                                 break;
450.
451.
                             b[y].shuffle();
452.
                             cout << "Bot shuffled cards of bot " << y << endl;</pre>
453.
                             z = 1;
454.
                             for (int j = 0; j < 3; j++) {
455.
                                  b[bn].bmem[y][j].rank = p.hand[j].rank;
456.
                                  b[bn].bmem[y][j].position = j;
457.
                                  b[bn].bmem[y][j].cond = 2;
458.
459.
                             for (int i = 0;i < 3;i++) {</pre>
460.
                                  if (i == bn) {
461.
                                      continue;
462.
                                  for (int j = 0; j < 3; j++) {
463.
464.
                                      b[i].bmem[y][j].position = 3;
465.
                                  }
466.
467.
                             break;
468.
                         }
469.
470.
471.
                         y = rand() % 4;
472.
                         x++;
473.
474.
                     if (z== 0) {
475.
                         while (y == bn) {
476.
                             y = rand() % 4;
477.
```

```
478.
                         switch (y) {
479.
                         case 3:
480.
481.
                              p.shuffle();
482.
                              cout << "Bot shuffled your cards" << endl;</pre>
483.
                              z = 1;
484.
                              for (int j = 0; j < 3; j++) {
485.
                                  b[bn].pmem[j].rank = p.hand[j].rank;
486.
                                  b[bn].pmem[j].position = j;
487.
                                  b[bn].pmem[j].cond = 2;
488.
489.
                              for (int i = 0; i < 3; i++) {
490.
                                  if (i == bn) {
491.
                                      continue;
492.
493.
                                  for (int j = 0; j < 3; j++) {
494.
                                      b[i].pmem[j].position = 3;
495.
496.
                              }
497.
498.
                              break;
499.
500.
                         default:
501.
502.
                              b[y].shuffle();
503.
                              cout << "Bot shuffled cards of bot " << y << endl;</pre>
504.
                              z = 1;
505.
                              for (int j = 0; j < 3; j++) {
506.
                                  b[bn].bmem[y][j].rank = p.hand[j].rank;
507.
                                  b[bn].bmem[y][j].position = j;
508.
                                  b[bn].bmem[y][j].cond = 2;
509.
510.
                              for (int i = 0;i < 3;i++) {</pre>
511.
                                  if (i == bn) {
512.
                                      continue;
513.
514.
                                  for (int j = 0; j < 3; j++) {
515.
                                      b[i].bmem[y][j].position = 3;
516.
517.
                              }
518.
                              break;
519.
520.
521.
522.
                     break;
523.
                 }
524.
                 case 7:
525.
526.
                     int plow = 4;
527.
                     int clow = 14;
528.
                     int cLown;
                     int myHighR = 0;
529.
```

```
530.
                     int myHighC = 4;
531.
                     bool found = 0;
532.
                     for (int i = 0;i < 3;i++) {</pre>
533.
                          if (b[bn].pmem[i].rank < clow &&</pre>
 b[bn].pmem[i].position == i && b[bn].pmem[i].cond>0) {
534.
                              plow = 3;
535.
                              clow = b[bn].pmem[i].rank;
536.
                              cLown = i;
537.
                              found = 1;
538.
539.
540.
                     for (int i = 0; i < 3; i++) {
541.
                          if (i == bn) {
542.
                              continue;
543.
544.
                          for (int j = 0; j < 3; j++) {
545.
                              if (b[bn].bmem[i][j].rank < clow &&</pre>
 b[bn].bmem[i][j].position == j && b[bn].bmem[i][j].cond>0) {
546.
                                  plow = i;
547.
                                  clow = b[bn].bmem[i][j].rank;
548.
                                  found = 1;
549.
                                  cLown = j;
550.
                              }
551.
552.
553.
                     if (found == 0) {
554.
                          for (int i = 0;i < 3;i++) {</pre>
555.
                              if (b[bn].pmem[i].rank < clow &&</pre>
 b[bn].pmem[i].position == 3 && b[bn].pmem[i].cond>0) {
556.
                                  plow = 3;
557.
                                  clow = b[bn].pmem[i].rank;
558.
                                  found = 1;
559.
                                  cLown = i;
560.
561.
562.
                          for (int i = 0;i < 3;i++) {</pre>
563.
                              if (i == bn) {
564.
                                  continue;
565.
566.
                              for (int j = 0; j < 3; j++) {
                                  if (b[bn].bmem[i][j].rank < clow &&</pre>
  b[bn].bmem[i][j].position == 3 && b[bn].bmem[i][j].cond>0) {
568.
                                       plow = i;
569.
                                       clow = b[bn].bmem[i][j].rank;
570.
                                       cLown = j;
571.
                                       found = 1;
572.
573.
574.
                          }
575.
576.
                     if (found == 0) {
577.
                          plow = rand() % 4;
```

```
578.
                         while (plow == bn) {
579.
                             plow = rand() % 4;
580.
581.
                         cLown = rand() % 4;
582.
583.
                    found = 0;
584.
                     for (int i = 0; i < 3; i++) {
585.
                         if (b[bn].bmem[bn][i].rank > myHighR &&
  b[bn].bmem[bn][i].cond > 0 && b[bn].bmem[bn][i].position == i) {
586.
                             myHighR = b[bn].bmem[bn][i].rank;
587.
                             myHighC = i;
588.
                             found = 1;
589.
590.
591.
                     if (found == 0) {
592.
                         for (int i = 0;i < 3;i++) {</pre>
                             if (b[bn].bmem[bn][i].rank > myHighR &&
  b[bn].bmem[bn][i].cond > 0 && b[bn].bmem[bn][i].position == 3) {
594.
                                 myHighR = b[bn].bmem[bn][i].rank;
595.
                                 myHighC = i;
596.
                                 found = 1;
597.
598.
599.
600.
                     if (found == 0) {
601.
                         myHighC = rand() % 4;
602.
603.
                    int tempr, tempmr, tempmc, tempmp;
604.
                    string temps;
605.
                    switch (plow) {
606.
                    case 3:
607.
608.
                         tempr = p.hand[cLown].rank;
609.
                         temps = p.hand[cLown].suit;
610.
                         p.hand[cLown].rank=b[bn].hand[myHighC].rank;
                         p.hand[cLown].suit = b[bn].hand[myHighC].suit;
611.
612.
                         b[bn].hand[myHighC].rank = tempr;
613.
                         b[bn].hand[myHighC].suit = temps;
614.
                         for (int i = 0;i < 3;i++) {</pre>
615.
                             tempmr = b[i].pmem[cLown].rank;
616.
                             tempmc = b[i].pmem[cLown].cond;
617.
                             tempmp = b[i].pmem[cLown].position;
                             b[i].pmem[cLown].rank =
  b[i].bmem[bn][myHighC].rank;
619.
                             b[i].pmem[cLown].cond =
  b[i].bmem[bn][myHighC].position;
620.
                             b[i].pmem[cLown].position =
  b[i].bmem[bn][myHighC].position;
621.
                             b[i].bmem[bn][myHighC].rank = tempmr;
622.
                             b[i].bmem[bn][myHighC].cond = tempmc;
623.
                             b[i].bmem[bn][myHighC].position = tempmp;
624.
```

```
625.
                         cout << "Bot exchnaged card with you" << endl;</pre>
626.
                         cout << "Your index: " << cLown << endl;</pre>
627.
                         cout << "Bot's index: " << myHighC << endl;</pre>
628.
                         break;
629.
                     }
630.
                     default:
631.
632.
                         tempr = b[plow].hand[cLown].rank;
633.
                         temps = b[plow].hand[cLown].suit;
634.
                         b[plow].hand[cLown].rank = b[bn].hand[myHighC].rank;
635.
                         b[plow].hand[cLown].suit = b[bn].hand[myHighC].suit;
636.
                         b[bn].hand[myHighC].rank = tempr;
                         b[bn].hand[myHighC].suit = temps;
637.
638.
                         for (int i = 0;i < 3;i++) {</pre>
639.
                              tempmr = b[i].bmem[plow][cLown].rank;
640.
                              tempmc = b[i].bmem[plow][cLown].cond;
641.
                              tempmp = b[i].bmem[plow][cLown].position;
642.
                              b[i].bmem[plow][cLown].rank =
  b[i].bmem[bn][myHighC].rank;
643.
                              b[i].bmem[plow][cLown].cond =
  b[i].bmem[bn][myHighC].position;
644.
                              b[i].bmem[plow][cLown].position =
  b[i].bmem[bn][myHighC].position;
645.
                              b[i].bmem[bn][myHighC].rank = tempmr;
646.
                              b[i].bmem[bn][myHighC].cond = tempmc;
647.
                              b[i].bmem[bn][myHighC].position = tempmp;
648.
                         cout << "Bot exchnaged card with bot "<<plow << endl;</pre>
649.
                         cout << "Bot "<<plow<< " index: " << cLown << endl;</pre>
650.
651.
                         cout << "Bot's index: " << myHighC << endl;</pre>
652.
                         break;
653.
654.
                     }
655.
                     break;
656.
                 }
657.
                 }
658.
            }
659.
660.
            void winner() {
                int sum[4];
661.
                 for (int i = 0; i < 3; i++) {
662.
663.
                     sum[i] = 0;
664.
                     for (int j = 0; j < 3; j++) {
                         sum[i] += b[i].hand[j].rank;
665.
666.
667.
                 }
668.
                 sum[3] = 0;
                 for (int i = 0;i < 3;i++) {</pre>
669.
670.
                     sum[3] += p.hand[i].rank;
671.
                 }
672.
                cout << endl;</pre>
673.
                 cout << "Your score: " << sum[3]<<endl;</pre>
```

```
674.
                for (int i = 0; i < 3; i++) {
675.
                    cout << "Bot " << i + 1 << " score: " << sum[i] << endl;</pre>
676.
677.
                cout << endl;</pre>
                if (sum[0] < sum[1] && sum[0] < sum[2] && sum[0] < sum[3]) {</pre>
678.
 //0
679.
                    cout << "BOT 1 WINS" << endl;</pre>
680.
                else if (sum[1] < sum[0] && sum[1] < sum[2] && sum[1] <
681.
  sum[3]) {
                   cout << "BOT 2 WINS" << endl;</pre>
683.
                else if (sum[2] < sum[0] && sum[2] < sum[3] && sum[2] <
684.
 sum[1]) {
               //2
                   cout << "BOT 3 WINS" << endl;</pre>
685.
686.
                else if (sum[3] < sum[0] && sum[3] < sum[1] && sum[3] <
 sum[2]) {
                   cout << "YOU WIN" << endl;</pre>
688.
689.
690.
                else if (sum[0] == sum[1] && sum[0] < sum[2] && sum[0] <
  sum[3]) { //01}
                   cout << "DRAW BETWEEN BOT 1 AND BOT 2" << endl;
691.
692.
693.
                else if (sum[0] == sum[2] && sum[0] < sum[3] && sum[0] <
  sum[1]) { //02}
                   cout << "DRAW BETWEEN BOT 1 AND BOT 3" << endl;
694.
695.
                else if (sum[0] == sum[3] && sum[0] < sum[2] && sum[0] <
  sum[1]) { //03}
697.
                   cout << "DRAW BETWEEN YOU AND BOT 1" << endl;
698.
699.
                else if (sum[1] == sum[2] && sum[1] < sum[3] && sum[1] <
  sum[0]) {
700.
                   cout << "DRAW BETWEEN BOT 2 AND BOT 3" << endl;</pre>
701.
702.
                else if (sum[1] == sum[3] && sum[1] < sum[0] && sum[1] <
 sum[2]) {
                //13
703.
                   cout << "DRAW BETWEEN YOU AND BOT 2" << endl;
704.
                else if (sum[2] == sum[3] && sum[2] < sum[0] && sum[2] <
  sum[1]) {
                //23
706.
                   cout << "DRAW BETWEEN YOU AND BOT 3" << endl;
707.
                else if (sum[0] == sum[1] \&\& sum[1] == sum[2] \&\& sum[2] <
708.
                //012
  sum[3]) {
709.
                   cout << "DRAW BETWEEN BOT 1, BOT 2 AND BOT 3" << endl;</pre>
710.
                else if (sum[0] == sum[1] \&\& sum[1] == sum[3] \&\& sum[1] <
711.
                //013
 sum[2]) {
712.
                   cout << "DRAW BETWEEN YOU, BOT 1 AND BOT 2" << endl;
713.
```

```
else if (sum[0] == sum[2] \&\& sum[2] == sum[3] \&\& sum[2] <
  sum[1]) {
                //023
715.
                    cout << "DRAW BETWEEN YOU, BOT 1 AND BOT 3" << endl;
716.
717.
                else if (sum[1] == sum[2] && sum[2] == sum[3] && sum[1] <
   sum[0]) {
                 //123
718.
                     cout << "DRAW BETWEEN YOU, BOT 2 AND BOT 3" << endl;
719.
720.
                else if (sum[1] == sum[2] && sum[2] == sum[3] && sum[3] ==
                 //123
                    cout << "DRAW BETWEEN ALL" << endl;</pre>
722.
723.
724.
           }
725.
726.
727.
       };
728.
729.
      int main()
730.
       {
731.
           srand(time(0));
732.
            cout << "WELCOME TO MIND BOGGLE" << endl << endl;</pre>
            cout << "RULES" << endl << "1) 3 cards will be distributed between 4
733.
   players, the player will look at those cards once and put them upside
   down.\n^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 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</pre>
  his 3 cards, look at them and shuffle them. \backslash n6) The game will end after
   all cards from deck are picked up.\n^7) Player whose sum of card rank is
   lowest will win." << endl;</pre>
735.
            Game q;
736.
            int x = 0;
737.
            g.initializeDeck();
738.
            q.distributeCards();
739.
            cout << "\nYour cards are:" << endl;</pre>
740.
            g.p.printHand();
741.
            for (int i = 0;i < 10;i++) {</pre>
742.
                cout << endl;</pre>
743.
                g.pmove(g.bank[x].rank, g.bank[x].suit);
744.
                x++;
745.
                for (int j = 0; j < 3; j++) {
746.
                    cout << endl;
747.
                    g.bmove(g.bank[x].rank, g.bank[x].suit, j);
748.
                    x++;
749.
                }
750.
751.
            g.printAllHands();
```

```
752. g.winner();
753.
754. }
```

10. OUTPUTS:

DIFFERENT THEMES

DIFFERENT THEMES
COLOR CODE BLACK Ø BLUE 1 AQUA 2 RED 3 PURPLE 4 Select your theme : 2
WELCOME TO MIND BOGGLE
COLOR CODE BLACK 0 BLUE 1 AQUA 2 RED 3 PURPLE 4
Select your theme : 3 NELCOME TO MIND BOGGLE
==================================
======================================

\$\$ \$\$\$\$_ \$\$_ _\$\$_ \$\$\$\$\$ \$\$\$\$\$ \$\$ \$\$\$ \$\$\$\$ \$\$\$\$ _ \$\$\$\$\$\$\$<u>_</u> \$\$\$\$\$\$ _\$\$__\$\$__ \$\$___\$\$_ _\$\$__\$\$ \$\$ \$\$_\$\$ _\$\$__\$\$_ \$\$_ \$\$\$\$\$\$\$ _\$\$_ \$\$_\$_\$\$_ \$\$_ \$\$_ \$\$\$\$\$_ \$\$_ \$\$__\$\$_ \$\$_ \$\$\$\$\$ \$\$_ _\$\$_ \$\$_ \$\$_ \$\$\$_ \$\$_ \$\$\$_ \$\$_ \$\$_\$\$\$\$<u> \$\$_</u> \$\$\$\$\$ \$\$_ \$\$___ _\$\$_ \$\$_ \$\$_ _\$\$__\$\$__\$\$ _\$\$__\$\$\$__\$\$ \$\$____ \$\$_\$\$\$_\$\$ \$<u>\$\$</u>\$\$_\$\$_\$\$ \$\$ _\$\$_ \$\$ _\$\$_ _\$\$_ _\$\$_ \$\$ \$\$ \$\$ __\$\$_ \$\$\$\$\$_ \$\$\$\$\$\$ \$\$\$\$\$\$\$ __\$\$\$ \$\$\$\$_ _\$\$\$\$_ \$\$\$\$\$\$\$ \$\$ \$\$ =======THEME========= CODE ______ BLACK a **BLUE**

PURPLE

Select your theme : 0 WELCOME TO MIND BOGGLE

AQUA

RED

Your cards are:

- 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 opponent「ÇÖs cards and shuffle them(now opponent doesn「ÇÖt know which card is wher
- 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 ΓÇϔΚingΓÇÖ, he can pick his 3 cards, look at them and shuffle them.

2

3

4

- 6) The game will end after all cards from deck are picked up.
- 7) Player whose sum of card rank is lowest will win.
- 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 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

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

```
You picked 8 of spades
Enter -1 to discard or Enter position of your card (0/1/2) to replace it
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
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
You Discarded 7 of clubs
Enter a bot number to exchnage card with
Enter a card number/positon to exchnage card
Enter your card number/positon to exchnage card
Bot 0 discarded Queen of spades without exchange Enter your card number/positon to exchange card
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!