LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2022/2023

PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA BRUTE FORCE

Disusun oleh :

Jimly Firdaus 13521102

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2023

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Permainan Kartu 24 biasa dimainkan dengan menggunakan kartu remi. Kartu remi terdiri dari 52 kartu yang terbagi menjadi empat suit (sekop, hati, keriting, dan wajik) yang masing-masing terdiri dari 13 kartu (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). Yang perlu diperhatikan hanyalah nilai kartu yang didapat (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). As bernilai 1, Jack bernilai 11, Queen bernilai 12, King bernilai 13, sedangkan kartu bilangan memiliki nilai dari bilangan itu sendiri. Pada awal permainan moderator atau salah satu pemain mengambil 4 kartu dari dek yang sudah dikocok secara random. Permainan berakhir ketika pemain berhasil menemukan solusi untuk membuat kumpulan nilainya menjadi 24. Pengubahan nilai tersebut dapat dilakukan menggunakan operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian (×), divisi (/) dan tanda kurung (()). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas.

1. Tulislah program kecil (sederhana) dalam Bahasa C/C++/Java yang mengimplementasikan algoritma Brute Force untuk mencari solusi word search puzzle.
2. Input: 4 angka/huruf yang terdiri dari:
   (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K). Contoh input:
   A 8 9 Q
   Selain itu, input juga dapat dilakukan dengan program men-generate 4 angka/hurufnya sendiri secara random. Pengguna dapat memilih apakah program meminta input dari pengguna atau generate sendiri. Apabila masukan tidak sesuai, maka program menampilkan luaran “Masukan tidak sesuai” dan akan meminta ulang.
3. Output:
   a. Banyak solusi yang ditemukan.

b. Solusi dari permainan kartu 24 ditampilkan di layar dan terdapat opsi untuk menyimpan solusi dalam file text. Untuk contoh kasus di atas A 8 9 Q, maka salah satu solusinya adalah:
   ((9 + A) - 8) * Q atau ((9 + 1) - 8) * 12

c. Waktu eksekusi program (tidak termasuk waktu pembacaan file input).

BAB II

PENJELASAN ALGORITMA *BRUTEFORCE* YANG DIGUNAKAN

Pertama-tama, akan dilakukan pembacaan *input* dari user melalui *console* dan divalidasi terlebih dahulu. Jika *input* sesuai / *valid*, maka akan dicari semua permutasi dari keempat huruf / angka yang di-*input* oleh user. Semua permutasi tersebut kemudian disimpan ke dalam sebuah array (*permutationArr*). Selanjutnya, akan dijelaskan algoritma untuk menyelesaikan permasalahan ini.

Setiap kombinasi dari ekspresi-ekspresi aritmatika yang dapat dibentuk dari 4 huruf / angka serta operasi aritmatika (/,*,+,-,()), terdapat pola sebagai berikut:

1. Dalam semua kemungkinan ekspresi terdapat pola ((a opr b) opr c) opr d.
2. Dalam semua kemungkinan ekspresi juga terdapat pola (a opr (b opr c)) opr d.
3. Dalam semua kemungkinan ekspresi juga terdapat pola (a opr b) opr (c opr d).
4. Dalam semua kemungkinan ekspresi juga terdapat pola a opr ((b opr c) opr d).
5. Dalam semua kemungkinan ekspresi juga terdapat pola a opr (b opr (c opr d)).
   Dengan a,b,c,d adalah angka / huruf dan opr adalah operator.

Dengan pola tersebut maka akan dimasukkan semua kemungkinan operator dengan kumpulan permutasi angka / huruf yang sudah didapat ke dalam bentuk pola tersebut. Setelah memasangkan semuanya, maka akan dihitung setiap ekspresi yang dibentuk dan setiap hasil yang sesuai (sama dengan 24) maka akan disimpan ke dalam *set* yang akan digunakan untuk menuliskan hasil semua solusi yang didapat.

BAB III

KODE PROGRAM DALAM BAHASA JAVA

```java
import java.util.Scanner;
import java.util.Arrays;
import java.util.ArrayList;
import java.util.Set;
import java.util.LinkedHashSet;
import java.time.Instant;
import java.time.Duration;
import java.util.Random;
import java.io.IOException;
import java.io.FileWriter;
import java.util.Collections;

Jimly-Firdaus
public class main {
    no usages
    private static final int target = 24;
    // Total combination of braces
    no usages
    private static final int total_method = 5;
    // File output path
    no usages
    private static final String pathname = "../test/";

    no usages
    private static Scanner scanner = new Scanner(System.in);
    Jimly-Firdaus
    public static void main (String[] args) throws IOException {
        splashScreen();
```

```java
        // Numbers container
        String[] numberChoiceCollection = new String[4];

        // Allowed card collection
        String[] cards = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

        // Prompt for input
        String choice;
        do{
            System.out.print("Random input ? (y/n): ");
            choice = scanner.next();
        } while (!choice.equals("y") && !choice.equals("n"));

        // Random numbers
        if (choice.equals("y")) {
            int min = 1, max = 13;
            Random random = new Random();
            numberChoiceCollection[0] = cards[random.nextInt( bound: max - min + 1)];
            numberChoiceCollection[1] = cards[random.nextInt( bound: max - min + 1)];
            numberChoiceCollection[2] = cards[random.nextInt( bound: max - min + 1)];
            numberChoiceCollection[3] = cards[random.nextInt( bound: max - min + 1)];
        }
        // Get numbers from console
        else {
            System.out.println("Enter 4 numbers (separated by single space): ");
            boolean validInput = false;
            while(!validInput) {
                for (int i = 0; i < numberChoiceCollection.length; i++) {
```

```
                if (i >= numberChoiceCollection.length){
                    break;
                }
                numberChoiceCollection[i] = scanner.next();
            }
            for (int i = 0; i < numberChoiceCollection.length; i++){
                if (Arrays.asList(cards).contains(numberChoiceCollection[i]) && i == numberChoiceCollection.length - 1) {
                    validInput = true;
                }
            }
            if (!validInput) {
                System.out.println("Please check your input");
            }
        }

    }
    // Show choosen cards
    System.out.println("Choosen cards: ");
    for (String number : numberChoiceCollection){
        System.out.print(number + " ");
    }
    System.out.println("");
    // Convert any allowed alphabet to number
    for (int i = 0; i < numberChoiceCollection.length; i++) {
        switch (numberChoiceCollection[i]){
            case "A":
                numberChoiceCollection[i] = "1";
                break;
```

```
            case "J":
                numberChoiceCollection[i] = "11";
                break;
            case "Q":
                numberChoiceCollection[i] = "12";
                break;
            case "K":
                numberChoiceCollection[i] = "13";
                break;
        }
    }

    // Map to new dynamic ArrayList to append all permutation of the 4 numbers
    ArrayList<String> numberChoiceArr = new ArrayList<>(Arrays.asList(numberChoiceCollection));
    ArrayList<ArrayList<String>> permutationArr = new ArrayList<>();
    permutations(numberChoiceArr, new ArrayList<String>(), permutationArr);

    // Solution container
    Set<String> answerSet = new LinkedHashSet<>();

    // Start Time
    Instant start = Instant.now();

    for (int i = 1; i <= total_method; i++) {
        for (ArrayList<String> numberChoice : permutationArr) {
            // Solve for every permutation of the 4 numbers
            solve(numberChoice.get(0), numberChoice.get(1), numberChoice.get(2), numberChoice.get(3), i, answerSet);
        }
```

```
        }

        // End timer
        Instant end = Instant.now();
        Duration timeElapsed = Duration.between(start, end);

        // Output choice
        do{
            System.out.print("Save solution to .txt ? (y/n): ");
            choice = scanner.next();
        } while (!choice.equals("y") && !choice.equals("n"));

        if (choice.equals("y")) {
            // Output to file
            outputFileHandler(answerSet);
        } else {
            // Output to console
            if (answerSet.size() == 0) {
                System.out.println("No Solutions");
            } else {
                System.out.println(answerSet.size() + " solutions found.");
                int index = 1;
                for (String element : answerSet) {
                    System.out.println(index + ". " + element);
                    index++;
                }
            }
        }
```

```
        }

        System.out.println("\nExecution Time: " + timeElapsed.toMillis() + " milliseconds");

    }

    /**
     * Give all the permutation of the given array
     * @param choosenNumArr given raw array
     * @param perm container for current permutation
     * @param permutationArr array containing all permutations
     * */
    no usages  Jimly-Firdaus
    public static void permutations(ArrayList<String> choosenNumArr, ArrayList<String> perm, ArrayList<ArrayList<String>>
     permutationArr) {
        if (choosenNumArr.size() == 0) {
            permutationArr.add(perm);
        } else {
            for (int i = 0; i < choosenNumArr.size(); i++) {
                ArrayList<String> newItems = new ArrayList<>(choosenNumArr);
                String newItem = newItems.remove(i);
                ArrayList<String> newPerm = new ArrayList<>(perm);
                newPerm.add(newItem);
                permutations(newItems, newPerm, permutationArr);
            }
        }
    }
```

```java
/**
 * Solve for 24 game
 * @param num_1, num_2, num_3, num_4 numbers
 * @param method combinations of braces
 * @param answerSet set containing all solutions
 * */
1 usage  Jimly-Firdaus
private static void solve (
        String num_1,
        String num_2,
        String num_3,
        String num_4,
        int method,
        Set<String> answerSet) {
    String[] operators = {"+", "-", "/", "*"};

    boolean generated_two_or_three_occurences = false;
    for (int i = 0; i < operators.length; i++) {
        String[] calculationResult = new String[2];
        String format = "", result = "";
        if (generated_two_or_three_occurences) {
            // for operators with no occurences
            boolean switchOpr = false;
            int position1 = i + 1 == operators.length ? 0 : i + 1;
            int position2 = position1 + 1 == operators.length ? 0 : position1 + 1;
            while (!switchOpr) {
                String pos1 = operators[i];
                String pos2 = operators[position1];
```

```
                    String pos3 = operators[position2];
                    calculation(num_1, num_2, num_3, num_4, pos1, pos2, pos3, method, calculationResult);
                    format = calculationResult[0];
                    result = calculationResult[1];
                    position2++;
                    if (position2 == operators.length) {
                        position1++;
                        position2 = 0;
                    }
                    if (Math.round(Double.parseDouble(result)) == target) {
                        answerSet.add(format);
                    }
                    if (position1 == operators.length) {
                        switchOpr = true;
                    }
                }

            }
            // for operators with 2-3 occurences
            else {
                int position = 1;
                int currentOpr = 0;
                int currentOpr2 = currentOpr;
                boolean changeOpr = false;
                while (!changeOpr) {
                    String pos1 = position == 1 ? operators[currentOpr] : operators[i];
                    String pos2 = position == 2 ? operators[currentOpr] : operators[i];
                    String pos3 = position == 3 ? operators[currentOpr] : operators[i];

                    calculation(num_1, num_2, num_3, num_4, pos1, pos2, pos3, method, calculationResult);
                    format = calculationResult[0];
                    result = calculationResult[1];

                    if (Math.round(Double.parseDouble(result)) == target) {
                        answerSet.add(format);
                    }
                    position++;
                    // if reached end of opr position
                    if (position == 4) {
                        // reset position & change opr
                        position = 1;
                        if (currentOpr == 3) {
                            changeOpr = true;
                        } else {
                            currentOpr++;
                        }
                        if (operators[i] == operators[3] && changeOpr == true) {
                            generated_two_or_three_occurences = true;
                            i = -1;
                        }
                    }
                }
            }
        }
    }

```

```java
/**
 * Calculate given numbers & operators with selected method
 * @param num_1, num_2, num_3, num_4 given numbers
 * @param pos1, pos2, pos3 given operators
 * @param method selected method
 * @param calculationResult array containing format & calculation result
 * */
2 usages  Jimly-Firdaus
private static void calculation (
        String num_1,
        String num_2,
        String num_3,
        String num_4,
        String pos1,
        String pos2,
        String pos3,
        int method,
        String[] calculationResult) {
    String format = "", result = "";
    switch (method) {
        case 1:
            format = "(" + "(" + num_1 + pos1 + num_2 + ")" + pos2 + num_3 + ")" + pos3 + num_4;
            result = evalExpr( fullExpr: evalExpr( fullExpr: evalExpr( fullExpr: num_1 + " " + pos1 + " " + num_2) + " " + pos2 + " "
              + num_3) + " " + pos3 + " " + num_4);
            break;
        case 2:
            format = "(" + num_1 + pos1 + "(" + num_2 + pos2 + num_3 + ")" + ")" + pos3 + num_4;
            result = evalExpr( fullExpr: evalExpr( fullExpr: num_1 + " " + pos1 + " " + evalExpr( fullExpr: num_2 + " " + pos2 + " "
              + num_3)) + " " + pos3 + " " + num_4);
            break;
        case 3:
            format = "(" + num_1 + pos1 + num_2 + ")" + pos2 + "(" + num_3 + pos3 + num_4 + ")";
            result = evalExpr( fullExpr: evalExpr( fullExpr: num_1 + " " + pos1 + " " + num_2) + " " + pos2 + " " +
             evalExpr( fullExpr: num_3 + " " + pos3 + " " + num_4));
            break;
        case 4:
            format = num_1 + pos1 + "(" + "(" + num_2 + pos2 + num_3 + ")" + pos3 + num_4 + ")";
            result = evalExpr( fullExpr: num_1 + " " + pos1 + " " + evalExpr( fullExpr: evalExpr( fullExpr: num_2 + " " + pos2 + " "
             + num_3) + " " + pos3 + " " + num_4));
            break;
        case 5:
            format = num_1 + pos1 + "(" + num_2 + pos2 + "(" + num_3 + pos3 + num_4 + ")" + ")";
            result = evalExpr( fullExpr: num_1 + " " + pos1 + " " + evalExpr( fullExpr: num_2 + " " + pos2 + " " +
             evalExpr( fullExpr: num_3 + " " + pos3 + " " + num_4)));
            break;
        default:
            format = "";
            result = "";
    }
    calculationResult[0] = format;
    calculationResult[1] = result;
}
```

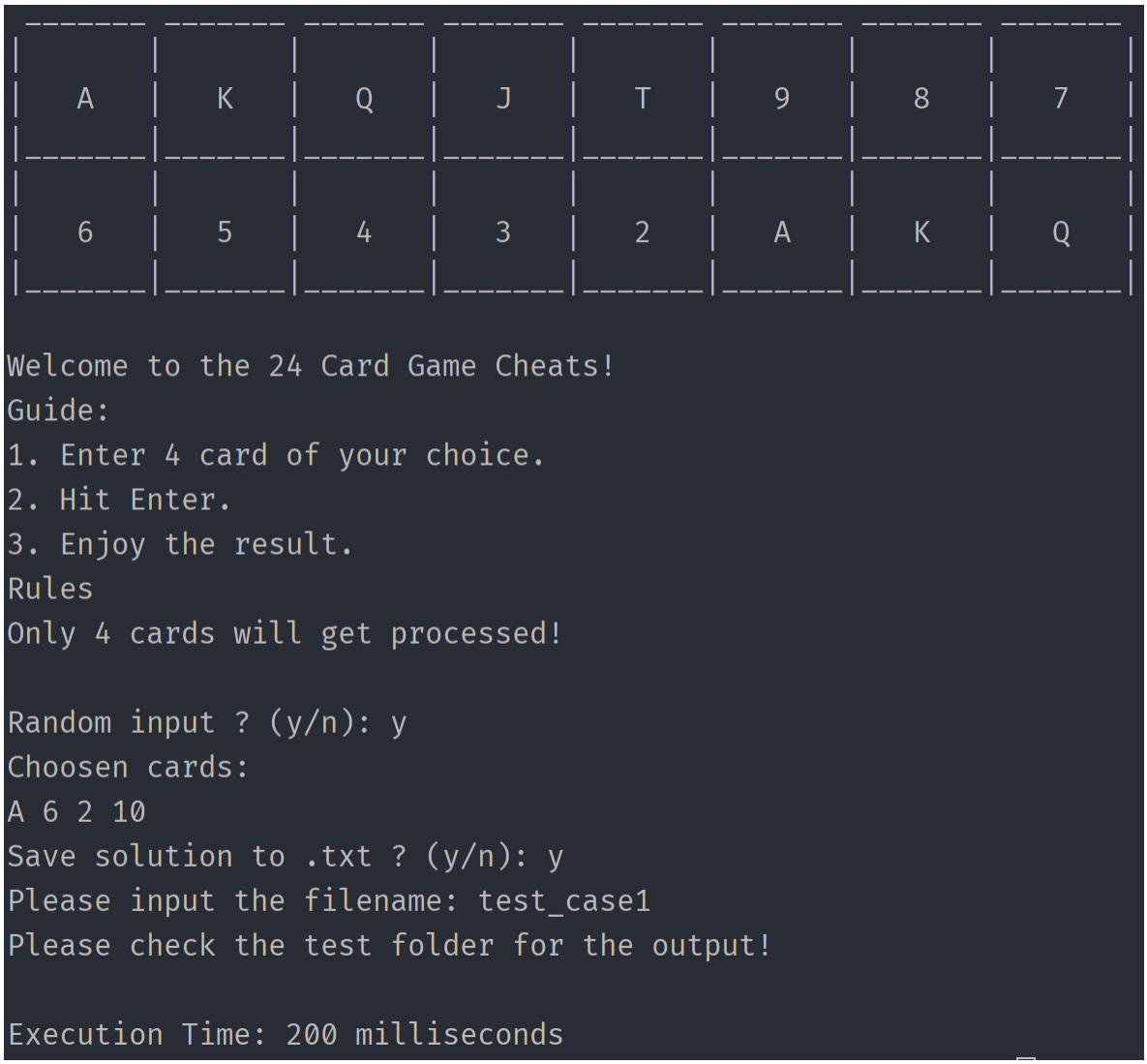
```java
/**
 * Arithmetic math evaluation
 * @param operator
 * @param num_1
 * @param num_2
 * @return arithmetic result in string
 * */
1 usage  Jimly-Firdaus
private static String eval (String operator, String num_1, String num_2) {
    double result = 0;
    double num1 = Double.parseDouble(num_1);
    double num2 = Double.parseDouble(num_2);
    String resultString;
    switch (operator){
        case "+":
            result = num1 + num2;
            break;
        case "-":
            result = num1 -  num2;
            break;
        case "*":
            result = num1 * num2;
            break;
        case "/":
            try {
                result = num1 / num2;
            } catch (ArithmeticException e) {
                result = 0;
            } catch (ArithmeticException e) {
                result = 0;
            }
            break;
    }
    resultString = Double.toString(result);
    return resultString;
}

/**
 * Takes in full math expression then pass to eval function
 * @param fullExpr math expression
 * @return arithmetic result in string
 * */
15 usages  Jimly-Firdaus
private static String evalExpr (String fullExpr) {
    String[] exprInArr = fullExpr.split( regex: " ");
    return eval(exprInArr[1], exprInArr[0], exprInArr[2]);
}
```

```java
    /**
     * File handler for output
     * @param answerSet set of all solutions
     * */
    1 usage   Jimly-Firdaus
    private static void outputFileHandler (Set<String> answerSet) throws IOException {
        System.out.print("Please input the filename: ");
        String fileName = scanner.next();
        String content = "";
        if (answerSet.size() != 0) {
            content += answerSet.size() + " solutions found\n";
            int index = 1;
            for (String element : answerSet) {
                content += index + ". " + element + "\n";
                index++;
            }
        } else {
            content = "No Solution\n";
        }
        FileWriter writer = new FileWriter ( fileName: pathname + fileName + ".txt");
        writer.write(content);
        writer.close();
        System.out.println("Please check the test folder for the output!");
    }

    no usages   Jimly-Firdaus
    public static void splashScreen() {
        System.out.println(" _______ _______ _______ _______ _______ _______ _______ _______ ");
        System.out.println("|       |       |       |       |       |       |       |       |");
        System.out.println("|   A   |   K   |   Q   |   J   |   T   |   9   |   8   |   7   |");
        System.out.println("|_______|_______|_______|_______|_______|_______|_______|_______|");
        System.out.println("|       |       |       |       |       |       |       |       |");
        System.out.println("|   6   |   5   |   4   |   3   |   2   |   A   |   K   |   Q   |");
        System.out.println("|_______|_______|_______|_______|_______|_______|_______|_______|");
        System.out.println();
        System.out.println("Welcome to the 24 Card Game Cheats!");
        System.out.println("Guide: ");
        System.out.println("1. Enter 4 card of your choice. ");
        System.out.println("2. Hit Enter. ");
        System.out.println("3. Enjoy the result. ");
        System.out.println("Rules");
        System.out.println("Only 4 cards will get processed!");
        System.out.println();
    }
}
```

BAB IV

INPUT / OUTPUT PROGRAM

Test Case 1 (Random: A 6 2 10)



Output

```
19 solutions found
1. ((1+6)*2)+10
2. ((1+2)*10)-6
3. ((6+1)*2)+10
4. ((2+1)*10)-6
5. ((10-1)*2)+6
6. (2*(1+6))+10
7. (2*(6+1))+10
8. (2*(10-1))+6
9. (10*(1+2))-6
10. (10*(2+1))-6
11. 6-((1-10)*2)
12. 6+((10-1)*2)
13. 6*((10/2)-1)
14. 10+((1+6)*2)
15. 10+((6+1)*2)
16. 6-(2*(1-10))
17. 6+(2*(10-1))
18. 10+(2*(1+6))
19. 10+(2*(6+1))
```

Test Case 2 (User Input: A A 8 8)

```
 _______ _______ _______ _______ _______ _______ _______ _______
|   A   |   K   |   Q   |   J   |   T   |   9   |   8   |   7   |
|_______|_______|_______|_______|_______|_______|_______|_______|
|       |       |       |       |       |       |       |       |
|   6   |   5   |   4   |   3   |   2   |   A   |   K   |   Q   |
|_______|_______|_______|_______|_______|_______|_______|_______|

Welcome to the 24 Card Game Cheats!
Guide:
1. Enter 4 card of your choice.
2. Hit Enter.
3. Enjoy the result.
Rules
Only 4 cards will get processed!

Random input ? (y/n): n
Enter 4 numbers (separated by single space):
A A 8 8
Choosen cards:
A A 8 8
Save solution to .txt ? (y/n): y
Please input the filename: test_case2
Please check the test folder for the output!

Execution Time: 169 milliseconds
```

Output

```
4 solutions found
1. ((1+1)*8)+8
2. (8*(1+1))+8
3. 8+((1+1)*8)
4. 8+(8*(1+1))
```

Test Case 3 (User Input: 1 8 9 Q)

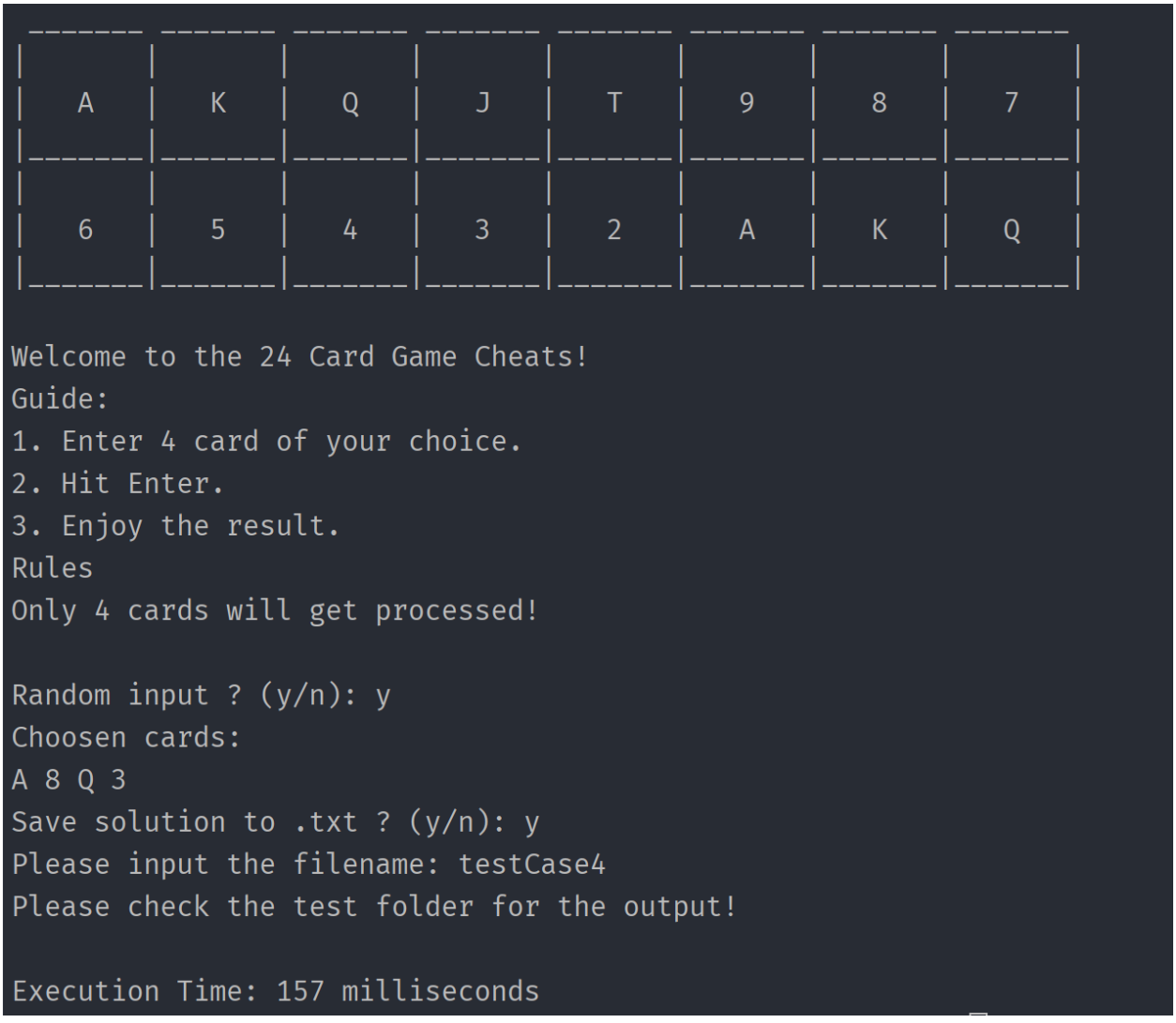
```
 _______ _______ _______ _______ _______ _______ _______ _______
|       |       |       |       |       |       |       |       |
|_______|_______|_______|_______|_______|_______|_______|_______|
|       |       |       |       |       |       |       |       |
|   6   |   5   |   4   |   3   |   2   |   A   |   K   |   Q   |
|_______|_______|_______|_______|_______|_______|_______|_______|

Welcome to the 24 Card Game Cheats!
Guide:
1. Enter 4 card of your choice.
2. Hit Enter.
3. Enjoy the result.
Rules
Only 4 cards will get processed!

Random input ? (y/n): n
Enter 4 numbers (separated by single space):
1 8 9 12
Please check your input
1 8 9 Q
Choosen cards:
1 8 9 Q
Save solution to .txt ? (y/n): y
Please input the filename: test_case3
Please check the test folder for the output!

Execution Time: 191 milliseconds
```

Output

```
46 solutions found
1. ((1+9)-8)*12
2. ((1*12)-9)*8
3. ((9+1)-8)*12
4. ((12*1)-9)*8
5. ((12-9)*1)*8
6. ((12-9)/1)*8
7. ((12-9)*8)*1
8. ((12-9)*8)/1
9. (1-(8-9))*12
10. (1+(9-8))*12
11. (1*(12-9))*8
12. (8*(12-9))*1
13. (8*(12-9))/1
14. (9+(1-8))*12
15. (9-(8-1))*12
16. (12-(1*9))*8
17. (12-(9*1))*8
18. (12-(9/1))*8
19. (1*8)*(12-9)
20. (8*1)*(12-9)
21. (8/1)*(12-9)
22. (12-9)/(1/8)
23. (12-9)*(1*8)
24. (12-9)*(8*1)
25. (12-9)*(8/1)
26. 1*((12-9)*8)
27. 8*((1*12)-9)
28. 8*((12*1)-9)
```

```
24. (12-9)*(8*1)
25. (12-9)*(8/1)
26. 1*((12-9)*8)
27. 8*((1*12)-9)
28. 8*((12*1)-9)
29. 8*((12/1)-9)
30. 8/((12/9)-1)
31. 8*((12-9)*1)
32. 8*((12-9)/1)
33. 12*((1-8)+9)
34. 12*((1+9)-8)
35. 12*((9+1)-8)
36. 12*((9-8)+1)
37. 1*(8*(12-9))
38. 8/(1/(12-9))
39. 8*(1*(12-9))
40. 8*(12-(1*9))
41. 8*(12-(9*1))
42. 8*(12-(9/1))
43. 12*(1-(8-9))
44. 12*(1+(9-8))
45. 12*(9+(1-8))
46. 12*(9-(8-1))
```

Test Case 4 (Random: A 8 Q 3)



Output

```
142 solutions found
1. ((1+8)+12)+3
2. ((1+8)+3)+12
3. ((1+12)+8)+3
4. ((1+12)+3)+8
5. ((1+3)+8)+12
6. ((1+3)+12)+8
7. ((8+1)+12)+3
8. ((8+1)+3)+12
9. ((8+12)+1)+3
10. ((8+12)+3)+1
11. ((8+3)+1)+12
12. ((8+3)+12)+1
13. ((12+1)+8)+3
14. ((12+1)+3)+8
15. ((12+8)+1)+3
16. ((12+8)+3)+1
17. ((12+3)+1)+8
18. ((12+3)+8)+1
19. ((3+1)+8)+12
20. ((3+1)+12)+8
21. ((3+8)+1)+12
22. ((3+8)+12)+1
23. ((3+12)+1)+8
24. ((3+12)+8)+1
25. (1+(8+12))+3
26. (1+(8+3))+12
27. (1+(12+8))+3
28. (1+(12+3))+8
29. (1+(3+8))+12
30. (1+(3+12))+8
31. (8+(1+12))+3
32. (8+(1/12))*3
33. (8-(1/12))*3
34. (8+(1+3))+12
35. (8+(12+1))+3
36. (8+(12+3))+1
37. (8+(3+1))+12
38. (8+(3+12))+1
39. (12+(1+8))+3
40. (12+(1+3))+8
41. (12+(8+1))+3
42. (12+(8+3))+1
43. (12+(3+1))+8
44. (12+(3+8))+1
45. (3+(1+8))+12
46. (3+(1+12))+8
47. (3+(8+1))+12
48. (3+(8+12))+1
49. (3+(12+1))+8
50. (3+(12+8))+1
51. (1+8)+(12+3)
52. (1+8)+(3+12)
53. (1+12)+(8+3)
54. (1+12)+(3+8)
55. (1+3)+(8+12)
56. (1+3)+(12+8)
```

57. (8+1)+(12+3)
58. (8+1)+(3+12)
59. (8+12)+(1+3)
60. (8*12)/(1+3)
61. (8+12)+(3+1)
62. (8*12)/(3+1)
63. (8+3)+(1+12)
64. (8*3)+(1/12)
65. (8*3)-(1/12)
66. (8+3)+(12+1)
67. (12+1)+(8+3)
68. (12+1)+(3+8)
69. (12+8)+(1+3)
70. (12*8)/(1+3)
71. (12+8)+(3+1)
72. (12*8)/(3+1)
73. (12+3)+(1+8)
74. (12+3)+(8+1)
75. (3+1)+(8+12)
76. (3+1)+(12+8)
77. (3+8)+(1+12)
78. (3*8)+(1/12)
79. (3*8)-(1/12)
80. (3+8)+(12+1)
81. (3+12)+(1+8)
82. (3+12)+(8+1)
83. 1+((8+12)+3)
84. 1+((8+3)+12)
85. 1+((12+8)+3)
86. 1+((12+3)+8)
87. 1+((3+8)+12)
88. 1+((3+12)+8)
89. 8+((1+12)+3)
90. 8+((1+3)+12)
91. 8/((1+3)/12)
92. 8+((12+1)+3)
93. 8+((12+3)+1)
94. 8*((12/3)-1)
95. 8+((3+1)+12)
96. 8/((3+1)/12)
97. 8+((3+12)+1)
98. 12+((1+8)+3)
99. 12+((1+3)+8)
100. 12/((1+3)/8)
101. 12+((8+1)+3)
102. 12+((8+3)+1)
103. 12+((3+1)+8)
104. 12/((3+1)/8)
105. 12+((3+8)+1)
106. 3+((1+8)+12)
107. 3+((1+12)+8)
108. 3*((1/12)+8)
109. 3+((8+1)+12)
110. 3+((8+12)+1)
111. 3+((12+1)+8)
112. 3+((12+8)+1)

```
112. 3+((12+8)+1)
113. 1+(8+(12+3))
114. 1+(8+(3+12))
115. 1+(12+(8+3))
116. 1+(12+(3+8))
117. 1+(3+(8+12))
118. 1+(3+(12+8))
119. 8+(1+(12+3))
120. 8+(1+(3+12))
121. 8+(12+(1+3))
122. 8*(12/(1+3))
123. 8+(12+(3+1))
124. 8*(12/(3+1))
125. 8+(3+(1+12))
126. 8+(3+(12+1))
127. 12+(1+(8+3))
128. 12+(1+(3+8))
129. 12+(8+(1+3))
130. 12*(8/(1+3))
131. 12+(8+(3+1))
132. 12*(8/(3+1))
133. 12+(3+(1+8))
134. 12+(3+(8+1))
135. 3+(1+(8+12))
136. 3+(1+(12+8))
137. 3+(8+(1+12))
138. 3*(8+(1/12))
139. 3*(8-(1/12))
140. 3+(8+(12+1))
```

```
120. 8+(1+(3+12))
121. 8+(12+(1+3))
122. 8*(12/(1+3))
123. 8+(12+(3+1))
124. 8*(12/(3+1))
125. 8+(3+(1+12))
126. 8+(3+(12+1))
127. 12+(1+(8+3))
128. 12+(1+(3+8))
129. 12+(8+(1+3))
130. 12*(8/(1+3))
131. 12+(8+(3+1))
132. 12*(8/(3+1))
133. 12+(3+(1+8))
134. 12+(3+(8+1))
135. 3+(1+(8+12))
136. 3+(1+(12+8))
137. 3+(8+(1+12))
138. 3*(8+(1/12))
139. 3*(8-(1/12))
140. 3+(8+(12+1))
141. 3+(12+(1+8))
142. 3+(12+(8+1))
```

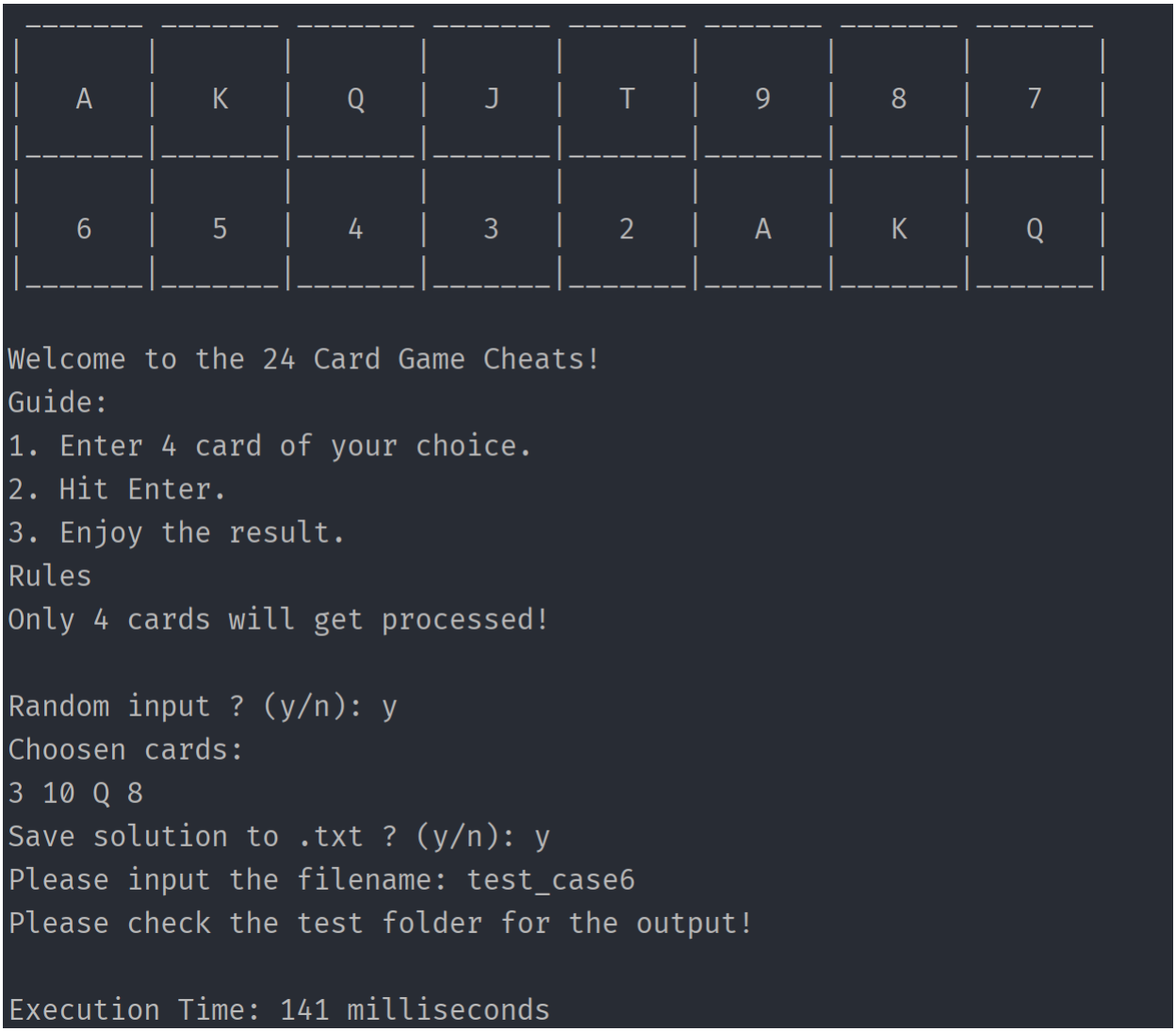
Test Case 5 (User Input: 3 3 3 3)

```
 _______ _______ _______ _______ _______ _______ _______ _______
|       |       |       |       |       |       |       |       |
|   A   |   K   |   Q   |   J   |   T   |   9   |   8   |   7   |
|_______|_______|_______|_______|_______|_______|_______|_______|
|       |       |       |       |       |       |       |       |
|   6   |   5   |   4   |   3   |   2   |   A   |   K   |   Q   |
|_______|_______|_______|_______|_______|_______|_______|_______|

Welcome to the 24 Card Game Cheats!
Guide:
1. Enter 4 card of your choice.
2. Hit Enter.
3. Enjoy the result.
Rules
Only 4 cards will get processed!

Random input ? (y/n): n
Enter 4 numbers (separated by single space):
3 3 3 3
Choosen cards:
3 3 3 3
Save solution to .txt ? (y/n): y
Please input the filename: test_case5
Please check the test folder for the output!

Execution Time: 137 milliseconds
```

Output

```
2 solutions found
1. ((3*3)*3)-3
2. (3*(3*3))-3
```

Test Case 6 (Random: 3 10 Q 8)



Output

```
6 solutions found
1. (10*12)/(8-3)
2. (12*10)/(8-3)
3. 10/((8-3)/12)
4. 12/((8-3)/10)
5. 10*(12/(8-3))
6. 12*(10/(8-3))
```

BAB V

TABEL PENILAIAN

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	√	
2. Program berhasil *running*	√	
3. Program dapat membaca input / generate sendiri dan memberikan luaran	√	
4. Solusi yang diberikan program memenuhi (berhasil mencapai 24)	√	
5. Program dapat menyimpan solusi dalam file teks	√	

Link Repository : https://github.com/Jimly-Firdaus/Tucil1_13521102