Tugas Kecil IF2211 Strategi Algoritma

Laporan Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*

Oleh:

Muhammad Bangkit Dwi Cahyono

K01 / 13521055

PROGRAM STUDIO TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

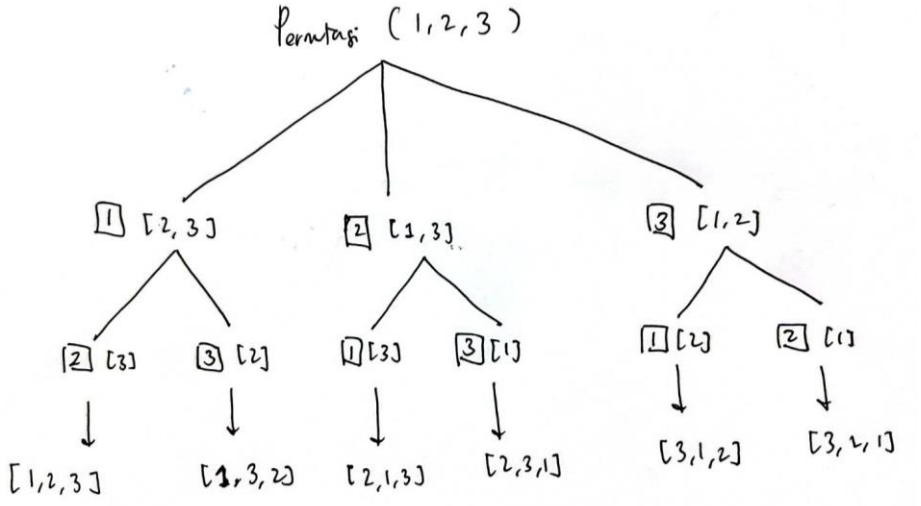
2023

A. Algoritma *Brute Force*

Algoritma *brute force* adalah sebuah pendekatan yang mudah untuk memecahkan suatu masalah, biasanya didasarkan langsung pada pernyataan masalah dan definisi konsep yang dilibatkan. Algoritma ini memecahkan masalah dengan sederhana, langsung (*straightforward*), dan dengan cara yang jelas. Algoritma ini menyelesaikan masalah dengan cara meninjau semua kasus yang muncul, lalu memilih kasus-kasus mana yang diinginkan program. Meskipun menyelesaikan suatu persoalan, tetap ada banyak cara yang berbeda dalam menyelesaikannya. Keragaman ini disebabkan pendekatan yang berbeda dalam mengerjakan persooalan tersebut, beberapa observasi yang dapat mengoptimalkan, dan membuang kasus yang tidak perlu.

Dalam menyelesaikan tugas kecil penyelesaian kartu 24 yang diberikan, digunakan algoritma *brute force*. Berikut langkah-langkah algoritma yang dibuat.

1. Pertama tentunya program akan membaca input dari pengguna untuk memilih apakah pengguna ingin memasukkan 4 kartu bebas atau 4 kartu bebas dipilih acak oleh program. Tentunya 4 kartu ini terdiri dari A, 2, 3, ..., J, Q, K, seperti halnya kartu remi. Kartu inilah yang nantinya dihitung semua kemungkinannya dengan menggunakan 4 operator yang ada (+, -, *, /) yang menghasilkan nilai 24. Tentunya input pengguna divalidasi oleh program.
2. Kemudian setelah 4 kartu sudah terpilih, saya melakukan permutasi kepada 4 angka tersebut. Algoritma yang digunakan adalah permutasi dengan rekursif sebagai berikut.



Mula-mula kita memiliki array yang berisi 4 angka, lalu kita bagi menjadi bagian-bagian kecil seperti halnya pada gambar di atas (dicontohkan 3 angka pada gambar). Diambil masing-masing elemen pada list dan sisa list yang tidak memiliki elemen tersebut. Lalu sisa list tadi direkursif ke fungsi awal, hasil dari rekursif itu kita *concat* ke elemen yang tadi di awal sudah kita pisah. Dengan demikian, didapatlah semua kemungkinan atau kombinasi dari 4 angka awal.

3. Setelah mendapatkan semua kemungkinan tadi, saatnya kita mengevaluasi dari kemungkinan yang ada dari tiap 4 angka tadi dengan menggunakan operator. Kita memiliki 4 angka, maka terdapat 3 kombinasi operator yang bisa kita gunakan dan ada 5 kemungkinan tanda kurung "(" dan ")" pada setiap hasil evaluasinya.
    a. (N OP N) OP (N OP N)
    b. ((N OP N) OP N) OP N
    c. (N OP (N OP N)) OP N
    d. N OP (N OP (N OP N))
    e. N OP ((N OP N) OP N)

    Dengan N adalah angka dan OP adalah operator. Algoritma yang dilakukan yaitu looping sebanyak 4 x 4 x 4 (karena ada kemungkinan 4 operator dalam 3 tempat), lalu diloop ke semua kombinasi hasil permutasi awal, yakni 24 disebabkan 4! = 24 kemungkinan. Dicoba semua kemungkinan tadi dengan 5 kasus tadi dengan "if" sebanyak 5 kali, lalu didapatlah hasil dari algoritma *brute force* tersebut.

4. Dikarenakan ada kemungkinan bahwa 4 kartu yang diinput di awal sama, misal "2 2 3 4" maka kita harus menghilangkan evaluasi dengan hasil yang sama, misal "3 * ((2' * 2'') + 4)" dengan "3 * ((2'' * 2') + 4)", karena solusi tersebut sama, dibedakan dengan penempatan angka 2 saja, maka hal tersebut dianggap 1 solusi saja. Untuk itu, digunakanlah struktur data *set* agar tidak ada data yang terduplikasi.
5. Setelah semua solusi didapat, ditampilkan banyaknya solusi dan tentunya solusi itu sendiri ke layar, dan juga waktu eksekusinya.
6. Setelah semua selesai, pengguna dapat men-*save* filenya jika ingin ke dalam filename.txt.

B. Source Program

```cpp
#include <iostream>
#include <float.h>

/* ADT and String Manipulation */
#include <sstream>
#include <string>
#include <vector>
#include <set>

/* Files */
#include <fstream>

/* UI and Time */
#include <ctime>
#include <windows.h>
#include <bits/stdc++.h>

using namespace std;
#define RESET   "\033[0m"
#define BLACK   "\033[1m\033[30m"      /* Black */
#define RED     "\033[1m\033[31m"      /* Red */
#define GREEN   "\033[1m\033[32m"      /* Green */
#define WHITE   "\033[1m\033[37m"      /* White */

int countWords(string str) {
    /* Menghitung banyaknya kata dalam 1 line of string (input user) */

    bool space = true;
    int count = 0;

    for (auto c:str) {
        if (isspace(c)) {
            space = true;
        } else {
            if (space) {
                ++count;
                space = false;
            }
        }
    }
    return count;
}
```

```cpp
bool isValid(string str) {
    /* Mengecek input user agar masukan harus [A, 2, ..., Q, K] */

    if (str.length() == 1) {
        /* Handle 1 char */
        if (!isdigit(str[0])) {
            if (str[0] != 'A' && str[0] != 'J' && str[0] != 'Q' && str[0] != 'K') {
                return false;
            }
        } else {
            if (str[0] == '0' or str[0] == '1') {
                return false;
            }
        }
    } else if (str.length() == 2) {
        /* Handle 2 char (10) */
        if (str[0] != '1' || str[1] != '0') {
            return false;
        }
    } else {
        return false;
    }

    return true;
}

bool isValid2(string s) {
    /* Mengecek input user agar masukan terdiri dari 4 masukan */

    if (countWords(s) != 4) {
        return false;
    }

    vector <string> input_splitted;
    string input2;
    stringstream ss(s);

    while (getline(ss, input2, ' ')) {
        input_splitted.push_back(input2);
    }
```

```
    if (isValid(input_splitted[0]) && isValid(input_splitted[1]) && isValid(input_splitted[2]) && isValid(input_splitted[3])) {
        return true;
    } else {
        return false;
    }
}

int strToNum(string str) {
    /* Mengubah string (masukan user) menjadi number */
    /* str pasti A, J, Q, K, atau angka 2 - 10 (sudah divalidasi) */

    if (str[0] == 'A') {
        return 1;
    } else if (str[0] == 'J') {
        return 11;
    } else if (str[0] == 'Q') {
        return 12;
    } else if (str[0] == 'K') {
        return 13;
    } else {
        if (str.length() == 2 && str[0] == '1' && str[1] == '0') {
            return 10;
        }
        return str[0] - '0';
    }
}

string numToString(int c) {
    /* Mengubah integer to string */

    switch(c) {
        case 1:
            return "A";
        case 11:
            return "J";
        case 12:
            return "Q";
        case 13:
            return "K";
        default:
            return to_string(c);
    }
}
```

```
double func(double a, char op, double b) {
    /* Menghitung operasi 2 angka dengan menggunakan 4 macam operator */

    switch(op) {
        case '+':
            return a + b;
        case '-':
            return a - b;
        case '*':
            return a * b;
        default:
            if (b != 0) {
                return a / b;
            } else {
                return -9999;
            }
    }
}

vector <vector<int>> permute(vector<int>& nums) {
    /* Menghasilkan vector of vector yang berisi hasil permutasi */

    if (nums.size() <= 1) { // basis
        return {nums};
    }

    vector <vector<int>> result;
    for (int i = 0; i < nums.size(); i ++) {
        vector<int> v(nums.begin(), nums.end());
        v.erase(v.begin() + i);
        auto res = permute(v); // rekurens

        for (int j = 0; j < res.size(); j ++) {
            vector<int> _v = res[j];
            _v.insert(_v.begin(), nums[i]);

            result.push_back(_v);
        }
    }

    return result;
}
```

```
int opToInt(char op) {
    /* Mengubah operator menjadi integer */

    switch(op) {
        case '+':
            return 0;
        case '-':
            return 1;
        case '*':
            return 2;
        default:
            return 3;
    }
}

double absolute(double x) {
    /* Menghitung absolute suatu angka */
    /* Handle error */

    if (x < 0) {
        return -x;
    } else {
        return x;
    }
}

vector <int> solver1() {
    /* Solver1 : user memasukkan input */

    string input, input2;
    vector <int> numbers;

    cout << "Enter 4 cards: ";
    getline(cin, input);

    while (!isValid2(input)) {
        cout << endl << RED << "Please enter 4 valid value [A, 2, ..., Q, K] !!!" << RESET << endl;
        cout << "Enter 4 cards: ";
        getline(cin, input);
    }
```

```
    stringstream ss(input);

    while (getline(ss, input2, ' ')) {
        numbers.push_back(strToNum(input2));
    }

    return numbers;
}

vector<int> solver2() {
    /* Solver2 : random generated number */
    /* Menghasilkan vektor dengan 4 random integer */

    srand(time(0));  // Initialize random number generator
    vector <int> random;

    for (int i = 0; i < 4; i ++) {
        random.push_back((rand() % 13) + 1); // 1 - 13
    }

    return random;
}

int inputOneTwo() {
    int n;

    while (true) {
        cout << "> ";
        if (cin >> n && (n == 1 || n == 2)) {
            break;
        } else {
            cout << RED << "Please enter a valid integer!! [1]/[2]" << RESET << endl;
            cin.clear();
        }
    }
    cin.ignore(80, '\n');

    return n;
}

void splash() {
    /* SPLASH SCREEN */
```

```
    cout << WHITE << ".------..------.       .------..------..------..------..------..------." << RESET << endl;
    cout << WHITE << "|" << RED << "2.---." << WHITE << " ||" << BLACK << "4.---. " << WHITE << "| " << RED << ".-.    " << WHITE << "|" << BLACK << "S.--. " << 
    cout << WHITE << "| " << RED << "(\\/)" << WHITE << " ||" << BLACK << " :/\\: " << WHITE << "|" << RED << "((5))   " << WHITE << "|" << BLACK << " :/\\:" 
    cout << WHITE << "| " << RED << ":\\/:" << WHITE << " ||" << BLACK << " :\\/: " << WHITE << "|" << RED << " '-.-.  " << WHITE << "|" << BLACK << " :\\/:" 
    cout << WHITE << "| " << RED << "'--'2" << WHITE << "|| " << BLACK << "'--'4" << WHITE << "|  " << RED << "((1)) " << WHITE << "|" << BLACK << " '--'S" << 
    cout << WHITE << "`------'`------'" << RED << "   '-'" << WHITE << "  `------'`------'`------'`------'`------'`------'" << RESET << endl << endl;
}

void command1() {
    /* LIST COMMAND 1 */

    cout << WHITE << "=======================================" << RESET << endl;
    cout << RED <<   "=             START/EXIT              =" << RESET << endl;
    cout << WHITE << "=======================================" << RESET << endl;
    cout << RED << "[1]" << WHITE << " START" << RESET << endl;
    cout << RED << "[2]" << WHITE << " EXIT" << RESET << endl;
}

void command2() {
    /* LIST COMMAND 2 */

    cout << WHITE << "=======================================" << RESET << endl;
    cout << RED <<   "=            List Command             =" << RESET << endl;
    cout << WHITE << "=======================================" << RESET << endl;
    cout << RED << "[1]" << WHITE << " Input 4 Cards [A, 2, ..., Q, K]" << RESET << endl;
    cout << RED << "[2]" << WHITE << " Generate 4 Random Cards" << RESET << endl;
}

void command3() {
    /* LIST COMMAND 3 */

    cout << endl << WHITE << "=======================================" << endl;
    cout << RED <<          "=       Do you want to save file?       =" << endl;
    cout << WHITE << "=======================================" << endl;
    cout << RED << "[1]" << WHITE <<" Yes" << endl;
    cout << RED << "[2]" << WHITE <<" No" << endl;
}

void app() {
    /* APP */

    int input;
```

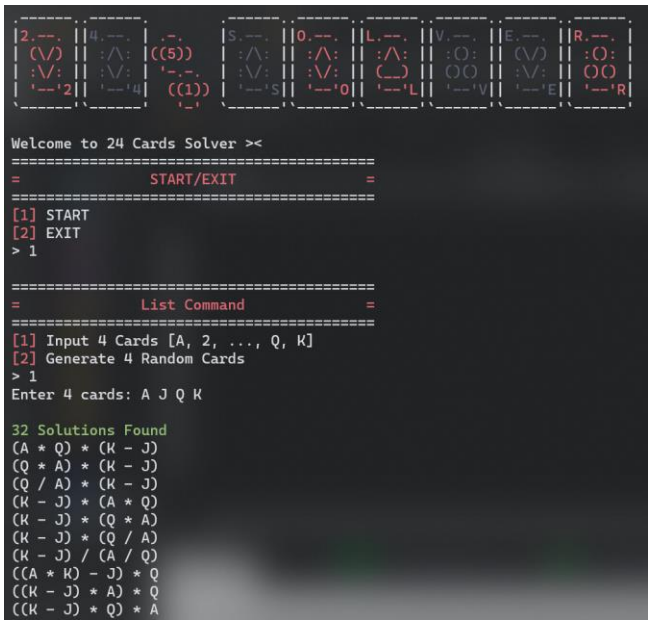
```
    int input;
    vector <int> numbers;

    cout << endl;
    command2();

    input = inputOneTwo();
    if (input == 1) {
        numbers = solver1();
    } else {
        numbers = solver2();
        cout << endl << GREEN << "Your cards are: " << RESET;
        for (auto x:numbers) {
            cout << numToString(x) << " ";
        }
        cout << endl;
    }

    /* Execution Time Init */
    clock_t start, end;
    start = clock();

    vector <vector<int>> permute_numbers = permute(numbers);
    vector <vector<int>> hasil;

    char operators[4] = { '+', '-', '*', '/' };

    for (int i = 0; i < 4; i ++) {
        for (int j = 0; j < 4; j ++) {
            for (int k = 0; k < 4; k ++) {
                for (int x = 0; x < 24; x ++) {
                    int a = permute_numbers[x][0];
                    int b = permute_numbers[x][1];
                    int c = permute_numbers[x][2];
                    int d = permute_numbers[x][3];

                    // (N OP N) OP (N OP N)
                    if (absolute(func(func(a, operators[i], b), operators[j], func(c, operators[k], d)) - 24) < FLT_EPSILON) {
                        hasil.push_back({0, a, opToInt(operators[i]), b, opToInt(operators[j]), c, opToInt(operators[k]), d});
                    }
```

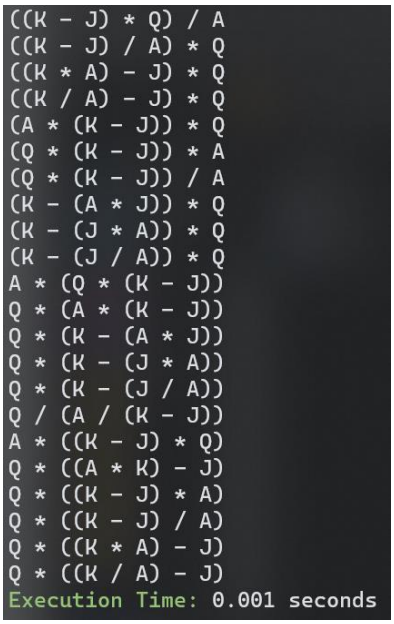
```
                    if (absolute(func(func(func(a, operators[i], b), operators[j], c), operators[k], d) - 24) < FLT_EPSILON) {
                        hasil.push_back({1, a, opToInt(operators[i]), b, opToInt(operators[j]), c, opToInt(operators[k]), d});
                    }

                    // (N OP (N OP N)) OP N
                    if (absolute(func(func(a, operators[i], func(b, operators[j], c)), operators[k], d) - 24) < FLT_EPSILON) {
                        hasil.push_back({2, a, opToInt(operators[i]), b, opToInt(operators[j]), c, opToInt(operators[k]), d});
                    }

                    // N OP (N OP (N OP N))
                    if (absolute(func(a, operators[i], func(b, operators[j], func(c, operators[k], d))) - 24) < FLT_EPSILON) {
                        hasil.push_back({3, a, opToInt(operators[i]), b, opToInt(operators[j]), c, opToInt(operators[k]), d});
                    }

                    // N OP ((N OP N) OP N)
                    if (absolute(func(a, operators[i], func(func(b, operators[j], c), operators[k], d)) - 24) < FLT_EPSILON) {
                        hasil.push_back({4, a, opToInt(operators[i]), b, opToInt(operators[j]), c, opToInt(operators[k]), d});
                    }
                }
            }
        }
    }

    set <vector<int>> s( hasil.begin(), hasil.end() );
    hasil.assign( s.begin(), s.end() );

    end = clock();

    if (s.size() != 0) {
        cout << endl << GREEN << s.size() << " Solutions Found" << RESET << endl;
    } else {
        cout << endl << GREEN << "No Solutions Found" << RESET << endl;
    }

    std::set< std::vector<int> >::iterator it;
    for (it = s.begin(); it != s.end(); it++) {
        const std::vector<int>& x = (*it);
        switch (x[0]) {
            case 0:
                cout << "(" << numToString(x[1]) << " " << operators[x[2]] << " " << numToString(x[3]) << ") " << operators[x[4]] << " (" << numToSt
                break;
            case 1:
                cout << "(("<< numToString(x[1]) << " " << operators[x[2]] << " " << numToString(x[3]) << ") " << operators[x[4]] << " " << numToSt
                break;
            case 2:
                cout << "(" << numToString(x[1]) << " " << operators[x[2]] << " (" << numToString(x[3]) << " " << operators[x[4]] << " " << numToSt
                break;
            case 3:
                cout << numToString(x[1]) << " " << operators[x[2]] << " " << "(" << numToString(x[3]) << " " << operators[x[4]] << " (" << numToSt
                break;
            case 4:
                cout << numToString(x[1]) << " " << operators[x[2]] << " ((" << numToString(x[3]) << " " << operators[x[4]] << " " << numToString(x
                break;
        }
    }

    double time_taken = double(end - start) / double(CLOCKS_PER_SEC);

    cout << GREEN << "Execution Time: " << RESET << time_taken << setprecision(6) << " seconds" << endl;

    if (s.size() != 0) {
        command3();
        int input2;

        input2 = inputOneTwo();

        if (input2 == 1) {
            string filename;

            cout << endl << "Input Filename: ";
            cin >> filename;

            ofstream MyFile("../test/" + filename + ".txt");

            // Write ke file
            MyFile << s.size() << " Solutions Found" << endl;

            std::set< std::vector<int> >::iterator itr;
            for (itr = s.begin(); itr != s.end(); itr++) {
                const std::vector<int>& x = (*itr);
                switch (x[0]) {
                    case 0:
                        MyFile << "(" << numToString(x[1]) << " " << operators[x[2]] << " " << numToString(x[3]) << ") " << operators[x[4]] << " ("
                        break;
```

```
                    case 0:
                        MyFile << "(" << numToString(x[1]) << " " << operators[x[2]] << " " << numToString(x[3]) << ") " << operators[x[4]] << " (" << numToStr
                        break;
                    case 1:
                        MyFile << "((" << numToString(x[1]) << " " << operators[x[2]] << " " << numToString(x[3]) << ") " << operators[x[4]] << " " << numToStri
                        break;
                    case 2:
                        MyFile << "(" << numToString(x[1]) << " " << operators[x[2]] << " (" << numToString(x[3]) << " " << operators[x[4]] << " " << numToStri
                        break;
                    case 3:
                        MyFile << numToString(x[1]) << " " << operators[x[2]] << " " << "(" << numToString(x[3]) << " " << operators[x[4]] << " (" << numToStri
                        break;
                    case 4:
                        MyFile << numToString(x[1]) << " " << operators[x[2]] << " ((" << numToString(x[3]) << " " << operators[x[4]] << " " << numToString(x[
                        break;
                }
            }

            // close file
            MyFile.close();

            cout << GREEN << "File saved succesfully" << RESET << endl;
        }
    }
}

int main () {
    /* Main Program */

    int startInput;

    splash();
    cout << "Welcome to 24 Cards Solver ><" << endl;
    command1();

    startInput = inputOneTwo();

    while (startInput == 1) {
        app();

        cout << endl << GREEN << "Do you want to try again?" << RESET << endl << endl;
        command1();
        startInput = inputOneTwo();
    }

    cout << endl << GREEN << "Thank You ><" << RESET << endl;
    exit(0);

    return 0;
}
```

C. Input dan Output

1. Testcase 1 (A J Q K)





```
==========================================
=       Do you want to save file?        =
==========================================
[1] Yes
[2] No
> 1

Input Filename: testcase1
File saved succesfully
```

```
32 Solutions Found
(A * Q) * (K - J)
(Q * A) * (K - J)
(Q / A) * (K - J)
(K - J) * (A * Q)
(K - J) * (Q * A)
(K - J) * (Q / A)
(K - J) / (A / Q)
((A * K) - J) * Q
((K - J) * A) * Q
((K - J) * Q) * A
((K - J) * Q) / A
((K - J) / A) * Q
((K * A) - J) * Q
((K / A) - J) * Q
(A * (K - J)) * Q
(Q * (K - J)) * A
(Q * (K - J)) / A
(K - (A * J)) * Q
(K - (J * A)) * Q
(K - (J / A)) * Q
A * (Q * (K - J))
Q * (A * (K - J))
Q * (K - (A * J))
Q * (K - (J * A))
Q * (K - (J / A))
Q / (A / (K - J))
A * ((K - J) * Q)
Q * ((A * K) - J)
Q * ((K - J) * A)
Q * ((K - J) / A)
Q * ((K * A) - J)
Q * ((K / A) - J)
```

(Hasil tetscase1.txt)

2. Testcase 2 (7 Q 6 K)

```
==========================================
=              List Command              =
==========================================
[1] Input 4 Cards [A, 2, ..., Q, K]
[2] Generate 4 Random Cards
> 1
Enter 4 cards: 7 Q 6 K

92 Solutions Found
(6 + Q) + (K - 7)
(6 + Q) - (7 - K)
(6 + K) + (Q - 7)
(6 + K) - (7 - Q)
(6 - 7) + (Q + K)
(6 - 7) + (K + Q)
(Q + 6) + (K - 7)
(Q + 6) - (7 - K)
(Q + K) + (6 - 7)
(Q + K) - (7 - 6)
(Q - 7) + (6 + K)
(Q - 7) + (K + 6)
(K + 6) + (Q - 7)
(K + 6) - (7 - Q)
(K + Q) + (6 - 7)
(K + Q) - (7 - 6)
(K - 7) + (6 + Q)
(K - 7) + (Q + 6)
((6 + Q) + K) - 7
((6 + Q) - 7) + K
```

```
((6 + K) + Q) - 7
((6 + K) - 7) + Q
((6 - 7) + Q) + K
((6 - 7) + K) + Q
((Q + 6) + K) - 7
((Q + 6) - 7) + K
((Q + K) + 6) - 7
((Q + K) - 7) + 6
((Q - 7) + 6) + K
((Q - 7) + K) + 6
((K + 6) + Q) - 7
((K + 6) - 7) + Q
((K + Q) + 6) - 7
((K + Q) - 7) + 6
((K - 7) + 6) + Q
((K - 7) + Q) + 6
((K - 7) * 6) - Q
(6 + (Q + K)) - 7
(6 + (Q - 7)) + K
(6 + (K + Q)) - 7
(6 + (K - 7)) + Q
(6 - (7 - Q)) + K
(6 - (7 - K)) + Q
(6 * (K - 7)) - Q
(Q + (6 + K)) - 7
(Q + (6 - 7)) + K
(Q + (K + 6)) - 7
(Q + (K - 7)) + 6
(Q - (7 - 6)) + K
(Q - (7 - K)) + 6
```

```
Q + (K + (6 - 7))
Q + (K - (7 - 6))
Q - (7 - (6 + K))
Q - (7 - (K + 6))
K + (6 + (Q - 7))
K + (6 - (7 - Q))
K + (Q + (6 - 7))
K + (Q - (7 - 6))
K - (7 - (6 + Q))
K - (7 - (Q + 6))
6 + ((Q + K) - 7)
6 + ((Q - 7) + K)
6 + ((K + Q) - 7)
6 + ((K - 7) + Q)
6 - ((7 - Q) - K)
6 - ((7 - K) - Q)
Q + ((6 + K) - 7)
Q + ((6 - 7) + K)
Q + ((K + 6) - 7)
Q + ((K - 7) + 6)
Q - ((7 - 6) - K)
Q - ((7 - K) - 6)
K + ((6 + Q) - 7)
K + ((6 - 7) + Q)
K + ((Q + 6) - 7)
K + ((Q - 7) + 6)
K - ((7 - 6) - Q)
K - ((7 - Q) - 6)
Execution Time: 0.001 seconds
```

```
==========================================
=       Do you want to save file?        =
==========================================
[1] Yes
[2] No
> 1

Input Filename: testcase2
File saved succesfully
```

```
92 Solutions Found
(6 + Q) + (K - 7)
(6 + Q) - (7 - K)
(6 + K) + (Q - 7)
(6 + K) - (7 - Q)
(6 - 7) + (Q + K)
(6 - 7) + (K + Q)
(Q + 6) + (K - 7)
(Q + 6) - (7 - K)
(Q + K) + (6 - 7)
(Q + K) - (7 - 6)
(Q - 7) + (6 + K)
(Q - 7) + (K + 6)
(K + 6) + (Q - 7)
(K + 6) - (7 - Q)
(K + Q) + (6 - 7)
(K + Q) - (7 - 6)
(K - 7) + (6 + Q)
(K - 7) + (Q + 6)
((6 + Q) + K) - 7
((6 + Q) - 7) + K
((6 + K) + Q) - 7
((6 + K) - 7) + Q
((6 - 7) + Q) + K
((6 - 7) + K) + Q
((Q + 6) + K) - 7
((Q + 6) - 7) + K
((Q + K) + 6) - 7
((Q + K) - 7) + 6
((Q - 7) + 6) + K
((Q - 7) + K) + 6
((K + 6) + Q) - 7
((K + 6) - 7) + Q
```

```
((K + Q) + 6) - 7
((K + Q) - 7) + 6
((K - 7) + 6) + Q
((K - 7) + Q) + 6
((K - 7) * 6) - Q
(6 + (Q + K)) - 7
(6 + (Q - 7)) + K
(6 + (K + Q)) - 7
(6 + (K - 7)) + Q
(6 - (7 - Q)) + K
(6 - (7 - K)) + Q
(6 * (K - 7)) - Q
(Q + (6 + K)) - 7
(Q + (6 - 7)) + K
(Q + (K + 6)) - 7
(Q + (K - 7)) + 6
(Q - (7 - 6)) + K
(Q - (7 - K)) + 6
(K + (6 + Q)) - 7
(K + (6 - 7)) + Q
(K + (Q + 6)) - 7
(K + (Q - 7)) + 6
(K - (7 - 6)) + Q
(K - (7 - Q)) + 6
6 + (Q + (K - 7))
6 + (Q - (7 - K))
6 + (K + (Q - 7))
6 + (K - (7 - Q))
6 - (7 - (Q + K))
6 - (7 - (K + Q))
Q + (6 + (K - 7))
Q + (6 - (7 - K))
Q + (K + (6 - 7))
```

```
Q + (K - (7 - 6))
Q - (7 - (6 + K))
Q - (7 - (K + 6))
K + (6 + (Q - 7))
K + (6 - (7 - Q))
K + (Q + (6 - 7))
K + (Q - (7 - 6))
K - (7 - (6 + Q))
K - (7 - (Q + 6))
6 + ((Q + K) - 7)
6 + ((Q - 7) + K)
6 + ((K + Q) - 7)
6 + ((K - 7) + Q)
6 - ((7 - Q) - K)
6 - ((7 - K) - Q)
Q + ((6 + K) - 7)
Q + ((6 - 7) + K)
Q + ((K + 6) - 7)
Q + ((K - 7) + 6)
Q - ((7 - 6) - K)
Q - ((7 - K) - 6)
K + ((6 + Q) - 7)
K + ((6 - 7) + Q)
K + ((Q + 6) - 7)
K + ((Q - 7) + 6)
K - ((7 - 6) - Q)
K - ((7 - Q) - 6)
```

(Hasil testcase2.txt)

3. Testcase 3 (A A A A)

```
==========================================
=              List Command              =
==========================================
[1] Input 4 Cards [A, 2, ..., Q, K]
[2] Generate 4 Random Cards
> 1
Enter 4 cards: A A A A

No Solutions Found
Execution Time: 0.001 seconds
```

4. Testcase 4 (Random) (2 2 5 A)

```
[2] Generate 4 Random Cards
> 2

Your cards are: 2 2 5 A

16 Solutions Found
(A + 5) * (2 + 2)
(A + 5) * (2 * 2)
(2 + 2) * (A + 5)
(2 + 2) * (5 + A)
(2 * 2) * (A + 5)
(2 * 2) * (5 + A)
(5 + A) * (2 + 2)
(5 + A) * (2 * 2)
((A + 5) * 2) * 2
((5 + A) * 2) * 2
(2 * (A + 5)) * 2
(2 * (5 + A)) * 2
2 * (2 * (A + 5))
2 * (2 * (5 + A))
2 * ((A + 5) * 2)
2 * ((5 + A) * 2)
Execution Time: 0.001 seconds
```

```
========================================
=       Do you want to save file?      =
========================================
[1] Yes
[2] No
> 1

Input Filename: testcase4
File saved succesfully
```

```
16 Solutions Found
(A + 5) * (2 + 2)
(A + 5) * (2 * 2)
(2 + 2) * (A + 5)
(2 + 2) * (5 + A)
(2 * 2) * (A + 5)
(2 * 2) * (5 + A)
(5 + A) * (2 + 2)
(5 + A) * (2 * 2)
((A + 5) * 2) * 2
((5 + A) * 2) * 2
(2 * (A + 5)) * 2
(2 * (5 + A)) * 2
2 * (2 * (A + 5))
2 * (2 * (5 + A))
2 * ((A + 5) * 2)
2 * ((5 + A) * 2)
```

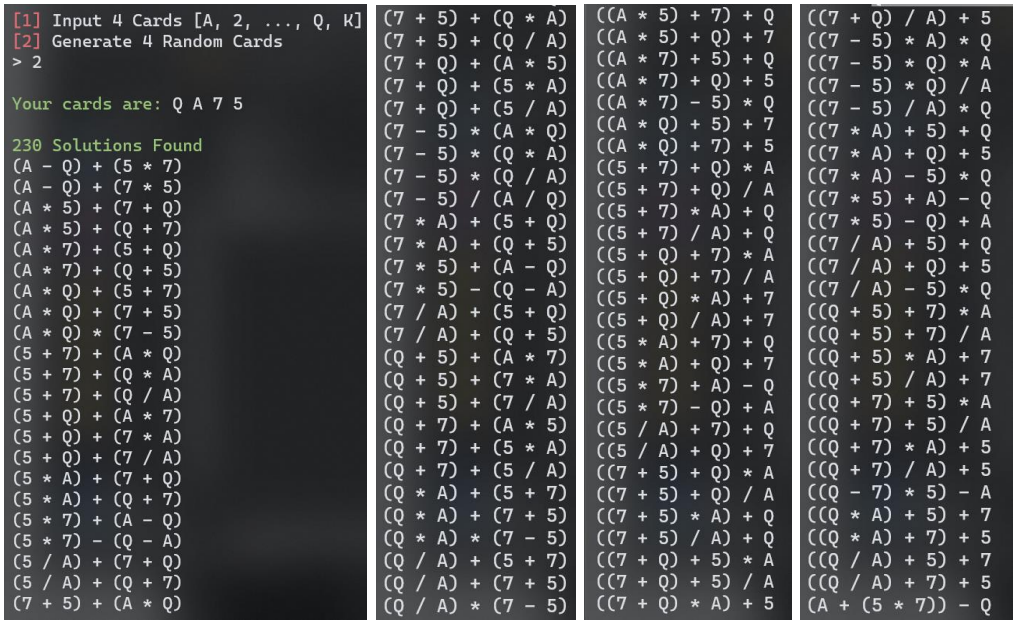
(Hasil testcase4.txt)

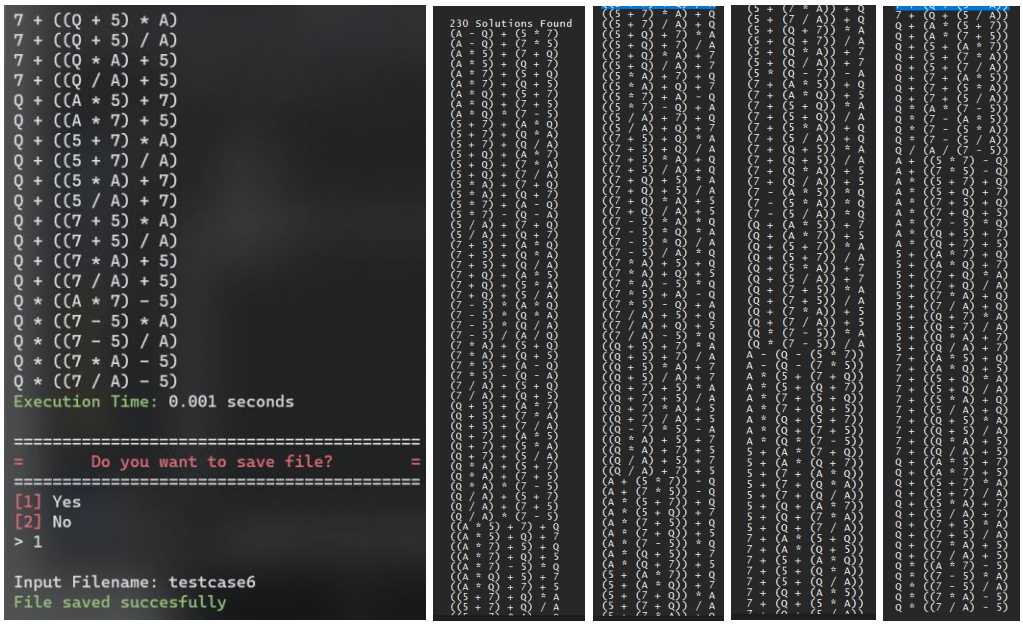
5. Testcase 5 (Random) (10 10 K 7)

```
========================================
=              List Command            =
========================================
[1] Input 4 Cards [A, 2, ..., Q, K]
[2] Generate 4 Random Cards
> 2

Your cards are: 10 10 K 7

No Solutions Found
Execution Time: 0 seconds
```

(Terkadang execution time 0 seconds)

6. Testcase 6 (Random) (Q A 7 5)


[1] Input 4 Cards [A, 2, ..., Q, K]
[2] Generate 4 Random Cards
> 2
Your cards are: Q A 7 5
230 Solutions Found
(A - Q) + (5 * 7)
(A - Q) + (7 * 5)
(A * 5) + (7 + Q)
(A * 5) + (Q + 7)
(A * 7) + (5 + Q)
(A * 7) + (Q + 5)
(A * Q) + (5 + 7)
(A * Q) + (7 + 5)
(A * Q) * (7 - 5)
(5 + 7) + (A * Q)
(5 + 7) + (Q * A)
(5 + 7) + (Q / A)
(5 + Q) + (A * 7)
(5 + Q) + (7 * A)
(5 + Q) + (7 / A)
(5 * A) + (7 + Q)
(5 * A) + (Q + 7)
(5 * 7) + (A - Q)
(5 * 7) - (Q - A)
(5 / A) + (7 + Q)
(5 / A) + (Q + 7)
(7 + 5) + (A * Q)
(7 + 5) + (Q * A)
(7 + 5) + (Q / A)
(7 + Q) + (A * 5)
(7 + Q) + (5 * A)
(7 + Q) + (5 / A)
(7 - 5) * (A * Q)
(7 - 5) * (Q * A)
(7 - 5) * (Q / A)
(7 - 5) / (A / Q)
(7 * A) + (5 + Q)
(7 * A) + (Q + 5)
(7 * 5) + (A - Q)
(7 * 5) - (Q - A)
(7 / A) + (5 + Q)
(7 / A) + (Q + 5)
(Q + 5) + (A * 7)
(Q + 5) + (7 * A)
(Q + 5) + (7 / A)
(Q + 7) + (A * 5)
(Q + 7) + (5 * A)
(Q + 7) + (5 / A)
(Q * A) + (5 + 7)
(Q * A) + (7 + 5)
(Q * A) * (7 - 5)
(Q / A) + (5 + 7)
(Q / A) + (7 + 5)
(Q / A) * (7 - 5)
((A * 5) + 7) + Q
((A * 5) + Q) + 7
((A * 7) + 5) + Q
((A * 7) + Q) + 5
((A * 7) - 5) * Q
((A * Q) + 5) + 7
((A * Q) + 7) + 5
((5 + 7) + Q) * A
((5 + 7) + Q) / A
((5 + 7) * A) + Q
((5 + 7) / A) + Q
((5 + Q) + 7) * A
((5 + Q) + 7) / A
((5 + Q) * A) + 7
((5 + Q) / A) + 7
((5 * A) + 7) + Q
((5 * A) + Q) + 7
((5 * 7) + A) - Q
((5 * 7) - Q) + A
((5 / A) + 7) + Q
((5 / A) + Q) + 7
((7 + 5) + Q) * A
((7 + 5) + Q) / A
((7 + 5) * A) + Q
((7 + 5) / A) + Q
((7 + Q) + 5) * A
((7 + Q) + 5) / A
((7 + Q) * A) + 5
((7 + Q) / A) + 5
((7 - 5) * A) * Q
((7 - 5) * Q) * A
((7 - 5) * Q) / A
((7 - 5) / A) * Q
((7 * A) + 5) + Q
((7 * A) + Q) + 5
((7 * A) - 5) * Q
((7 * 5) + A) - Q
((7 * 5) - Q) + A
((7 / A) + 5) + Q
((7 / A) + Q) + 5
((7 / A) - 5) * Q
((Q + 5) + 7) * A
((Q + 5) + 7) / A
((Q + 5) * A) + 7
((Q + 5) / A) + 7
((Q + 7) + 5) * A
((Q + 7) + 5) / A
((Q + 7) * A) + 5
((Q + 7) / A) + 5
((Q - 7) * 5) - A
((Q * A) + 5) + 7
((Q * A) + 7) + 5
((Q / A) + 5) + 7
((Q / A) + 7) + 5
(A + (5 * 7)) - Q
(A + (5 * 7)) + Q
(A + (7 * 5)) - Q
(A * (5 + 7)) + Q
(A * (5 + Q)) + 7
(A * (7 + 5)) + Q
(A * (7 + Q)) + 5
(A * (7 - 5)) * Q
(A * (Q + 5)) + 7
(A * (Q + 7)) + 5
(5 + (A * 7)) + Q
(5 + (A * Q)) + 7
(5 + (7 + Q)) * A
(5 + (7 + Q)) / A
(5 + (7 * A)) + Q
(5 + (7 / A)) + Q
(5 + (Q + 7)) * A
(5 + (Q + 7)) / A
(5 + (Q * A)) + 7
(5 + (Q / A)) + 7
(5 * (Q - 7)) - A
(7 + (A * 5)) + Q
(7 + (A * Q)) + 5
(7 + (5 + Q)) * A
(7 + (5 + Q)) / A
(7 + (5 * A)) + Q
(7 + (5 / A)) + Q
(7 + (Q + 5)) * A
(7 + (Q + 5)) / A
(7 + (Q * A)) + 5
(7 + (Q / A)) + 5
(7 - (A * 5)) * Q
(7 - (5 * A)) * Q
(7 - (5 / A)) * Q
(Q + (A * 5)) + 7
(Q + (A * 7)) + 5
(Q + (5 + 7)) * A
(Q + (5 + 7)) / A
(Q + (5 * A)) + 7
(Q + (5 / A)) + 7
(Q + (7 + 5)) * A
(Q + (7 + 5)) / A
(Q + (7 * A)) + 5
(Q + (7 / A)) + 5
(Q * (7 - 5)) * A
(Q * (7 - 5)) / A
A - (Q - (5 * 7))
A - (Q - (7 * 5))
A * (5 + (7 + Q))
A * (5 + (Q + 7))
A * (7 + (5 + Q))
A * (7 + (Q + 5))
A * (Q + (5 + 7))
A * (Q + (7 + 5))
A * (Q * (7 - 5))
5 + (A * (7 + Q))
5 + (A * (Q + 7))
5 + (7 + (A * Q))
5 + (7 + (Q * A))
5 + (7 + (Q / A))
5 + (Q + (A * 7))
5 + (Q + (7 * A))
5 + (Q + (7 / A))
7 + (A * (5 + Q))
7 + (A * (Q + 5))
7 + (5 + (A * Q))
7 + (5 + (Q * A))
7 + (5 + (Q / A))
7 + (Q + (A * 5))
7 + (Q + (5 * A))
7 + (Q + (5 / A))
Q + (A * (5 + 7))
Q + (A * (7 + 5))
Q + (5 + (A * 7))
Q + (5 + (7 * A))
Q + (5 + (7 / A))
Q + (7 + (A * 5))
Q + (7 + (5 * A))
Q + (7 + (5 / A))
Q * (A * (7 - 5))
Q * (7 - (A * 5))
Q * (7 - (5 * A))
Q * (7 - (5 / A))
Q / (A / (7 - 5))
A + ((5 * 7) - Q)
A + ((7 * 5) - Q)
A * ((5 + 7) + Q)
A * ((5 + Q) + 7)
A * ((7 + 5) + Q)
A * ((7 + Q) + 5)
A * ((7 - 5) * Q)
A * ((Q + 5) + 7)
A * ((Q + 7) + 5)
5 + ((A * 7) + Q)
5 + ((A * Q) + 7)
5 + ((7 + Q) * A)
5 + ((7 + Q) / A)
5 + ((7 * A) + Q)
5 + ((7 / A) + Q)
5 + ((Q + 7) * A)
5 + ((Q + 7) / A)
5 + ((Q * A) + 7)
5 + ((Q / A) + 7)
7 + ((A * 5) + Q)
7 + ((A * Q) + 5)
7 + ((5 + Q) * A)
7 + ((5 + Q) / A)
7 + ((5 * A) + Q)
7 + ((5 / A) + Q)
7 + ((Q + 5) * A)



(Hasil testcase6.txt)

D. *Link To Repository*

https://github.com/bangkitdc/Tucil1_13521055

E. Table

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	✓	