Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II Tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun Oleh :

Johanes Lee

13521148

Kelas 2

Program Studi S1 Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2023

Bab I

Deskripsi Persoalan

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan berhitung serta mengasah otak agar dapat berpikir dengan cepat dan akurat. 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. (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf).

Bab II

Algoritma Penyelesaian

Berikut merupakan penyelesaian persoalan permainan yang sudah dideskripsikan dengan pendekatan *brute force*.

1. Cari semua kombinasi susunan 3 operator (dengan juga memperhatikan masing-masing permutasi atau urutannya) yang mungkin diperoleh dari operator tambah (+), kurang (-), kali (*), dan bagi (/). Contohnya +++, +*-, -+*, dan /-*.

2. Cari semua permutasi urutan 4 kartu yang diperoleh.

3. Untuk setiap urutan permutasi kartu yang diperoleh, lakukan hal berikut.

    3.1 Untuk setiap urutan operator yang mungkin disisipkan di antara 4 bilangan kartu, lakukan hal berikut.

        3.1.1 Untuk setiap susunan tanda kurung yang mungkin (merepresentasikan semua kemungkinan urutan evaluasi ekspresi) untuk ekspresi aritmatika 4 operan dan 3 operator: ((a b) c) d; (a b) (c d); (a (b c)) d; a ((b c) d); dan a (b (c d)), lakukan hal berikut.

            a) Hitung hasil ekspresi aritmatika yang telah terbentuk dengan memperhatikan susunan tanda kurung dalam menentukan prioritas operasi yang perlu didahulukan.
            b) Jika ekspresi aritmatika tersebut menghasilkan angka 24, masukkan ekspresi tersebut sebagai salah satu solusi.

Bab III

Implementasi Program

Berikut adalah implementasi program untuk mencari semua solusi persoalan permainan kartu 24 dalam bahasa C++ menggunakan algoritma *brute force*.

3.1 card.cpp

```
using namespace std;
#include <sstream>
#include <cstring>
#include <iostream>
#include <regex>

#include "./headers/card.h"

map<string, int> cardMap {{"A", 1}, {"2", 2}, {"3", 3}, {"4", 4}, {"5", 5}, {"6",
6}, {"7", 7}, {"8", 8}, {"9", 9}, {"10", 10}, {"J", 11}, {"Q", 12}, {"K", 13}};
map<int, string> numMap {{1, "A"}, {2, "2"}, {3, "3"}, {4, "4"}, {5, "5"}, {6,
"6"}, {7, "7"}, {8, "8"}, {9, "9"}, {10, "10"}, {11, "J"}, {12, "Q"}, {13, "K"}};

vector<string> split(string s, char delimeter)
{
    // split string into vector of strings by certain delimeter

    vector<string> res;
    string substr;
    istringstream stream(s);

    while (getline(stream, substr, delimeter)) {
        res.push_back(substr);
    }

    return res;
}

bool isCardString(string s)
{
    // mengecek apakah karakater-karakter yang digunakan valid

    int idx = strspn(s.c_str(), "0123456789AJQK ");

    return idx == s.length();
}

bool isNumCardsValid(vector<int> input)
{
    // return true if input is 4 cards with valid value range

    int l = input.size();

    if (l != 4)
    {
        cout << "Input harus berupa 4 karakter kartu valid yang dipisahi
spasi!\n";
        return false;
    }

    for (int i = 0; i < l; i++)
    {
```

```
        if(input[i] <= 0 || input[i] >= 14)
        {
            cout << "Input kartu hanya dapat berupa angka [2, 10] atau karakter
A, J, Q, dan K (huruf kapital)!\n";
            return false;
        }
    }

    return true;

}

vector<int> cardToNum(vector<string> cards)
{
    // mengubah vektor string kartu menjadi vektor nilai kartu tersebut

    vector<int> res;

    int l = cards.size();

    for(int i = 0; i < l; i++)
    {
        res.push_back(cardMap[cards[i]]);
    }

    return res;

}

vector<string> numToCard(vector<int> nums)
{
    // mengubah vektor nilai kartu menjadi vektor string kartu

    vector<string> res;

    int l = nums.size();

    for(int i = 0; i < l; i++)
    {
        res.push_back(numMap[nums[i]]);
    }

    return res;

}
```

3.2 game.cpp

```
using namespace std;

#include <regex>
#include <float.h>

// from src
#include "./headers/game.h"
#include "./headers/card.h"

vector<string> opCombination;
bool isConfigured = false;

float float_abs(float a)
{
    if (a < 0)
    {
        return -a;
    }
```

```
    return a;
}

vector<int> generateRandom()
{
    // generate 4 random numbers in a vector
    vector<int> res{0, 0, 0, 0};

    for (int i = 0; i < 4; i++)
    {
        res[i] = (rand() % 13) + 1;
    }

    return res;
}

set<vector<int>> permute(vector<int> nums, int offset, int idx)
{
    // return all permutation of nums

    set<vector<int>> res {};

    if (idx >= offset)
    {
        return res;
    }

    if (idx == offset - 1)
    {
        res.insert(nums);

        return res;
    }

    for (int i = idx; i < offset; i++)
    {
        swap(nums[idx], nums[i]);

        set<vector<int>> temp = permute(nums, offset, idx+1);

        set_union(res.begin(), res.end(), temp.begin(), temp.end(), inserter(res,
res.begin()));

        swap(nums[idx], nums[i]);
    }

    return res;
}

vector<string> repeatingCombination(string symbols, int symbolNum, int length)
{
    // return repeating combination (and their permutations) from distinct symbols

    vector<string> res {};

    if (length == 0)
    {
        return res;
    }

    if (length == 1)
    {
        for (int i = 0; i < symbolNum; i++)
        {
            res.push_back(symbols.substr(i, 1));
        }

        return res;
    }

    vector<string> rec = repeatingCombination(symbols, symbolNum, length-1);

    for (int i = 0; i < symbolNum; i++)
    {
        vector<string> temp = rec;
        int l = temp.size();
```

```
        for (int j = 0; j < l; j++)
        {
            temp[j] = symbols.substr(i, 1) + temp[j];
        }

        res.insert(res.end(), temp.begin(), temp.end());
    }

    return res;
}

float calculate(float a, float b, char op)
{
    // menghitung hasil operasi a dan b dengan operator op

    if (op == '+')
    {
        return a + b;
    }

    else if (op == '-')
    {
        return a - b;
    }

    else if (op == '*')
    {
        return a * b;
    }

    else if (op == '/')
    {
        return a / b;
    }

    return 0;
}

float evaluator(vector<int>nums, string ops,  int parenthesisMode)
{
    // mengevaluasi hasil ekspresi matematika dengan 4 bilangan dan 3 operator
    // parenthesisMode == 0 -> ((a b) c) d
    // parenthesisMode == 1 -> (a b) (c d)
    // parenthesisMode == 2 -> (a (b c)) d
    // parenthesisMode == 3 -> a ((b c) d)
    // parenthesisMode == 4 -> a (b (c d))

    float res;

    if (parenthesisMode == 0)
    {
        float temp1 = calculate(nums[0], nums[1], ops[0]);
        float temp2 = calculate(temp1, nums[2], ops[1]);

        res = calculate(temp2, nums[3], ops[2]);
    }

    else if (parenthesisMode == 1)
    {
        float temp1 = calculate(nums[0], nums[1], ops[0]);
        float temp2 = calculate(nums[2], nums[3], ops[2]);

        res = calculate(temp1, temp2, ops[1]);
    }

    else if (parenthesisMode == 2)
    {
        float temp1 = calculate(nums[1], nums[2], ops[1]);
        float temp2 = calculate(nums[0], temp1, ops[0]);

        res = calculate(temp2, nums[3], ops[2]);
    }

    else if (parenthesisMode == 3)
    {
```

```
        float temp1 = calculate(nums[1], nums[2], ops[1]);
        float temp2 = calculate(temp1, nums[3], ops[2]);

        res = calculate(nums[0], temp2, ops[0]);
    }

    else
    {
        float temp1 = calculate(nums[2], nums[3], ops[2]);
        float temp2 = calculate(nums[1], temp1, ops[1]);

        res = calculate(nums[0], temp2, ops[0]);
    }

    return res;

}

string splitOp(string op)
{
    return " " + op + " ";
}

string generateString (vector<int>nums, string ops,  int parenthesisMode)
{
    // mengevaluasi hasil ekspresi matematika dengan 4 bilangan dan 3 operator
    // parenthesisMode == 0 -> ((a b) c) d
    // parenthesisMode == 1 -> (a b) (c d)
    // parenthesisMode == 2 -> (a (b c)) d
    // parenthesisMode == 3 -> a ((b c) d)
    // parenthesisMode == 4 -> a (b (c d))

    vector<string> cards = numToCard(nums);

    string res = "";

    if (parenthesisMode == 0)
    {
        res = "((" + cards[0] + splitOp(ops.substr(0, 1)) + cards[1] + ")" +
splitOp(ops.substr(1, 1)) + cards[2] + ")" + splitOp(ops.substr(2, 1)) + cards[3];
    }

    else if (parenthesisMode == 1)
    {
        res = "(" + cards[0] + splitOp(ops.substr(0, 1)) + cards[1] + ")" +
splitOp(ops.substr(1, 1)) + "(" + cards[2] + splitOp(ops.substr(2, 1)) + cards[3] + ")";
    }

    else if (parenthesisMode == 2)
    {
        res = "(" + cards[0] + splitOp(ops.substr(0, 1)) + "(" + cards[1] +
splitOp(ops.substr(1, 1)) + cards[2] + "))" + splitOp(ops.substr(2, 1)) + cards[3];
    }

    else if (parenthesisMode == 3)
    {
        res = cards[0] + splitOp(ops.substr(0, 1)) + "((" + cards[1]  +
splitOp(ops.substr(1, 1)) + cards[2] + ")" + splitOp(ops.substr(2, 1)) + cards[3] + ")";
    }

    else
    {
        res = cards[0] + splitOp(ops.substr(0, 1)) + "(" + cards[1] + splitOp(ops.substr(1,
1)) + "(" + cards[2] + splitOp(ops.substr(2, 1)) + cards[3] + "))";
    }

    return res;

}

void config()
{
    // membuat konfigurasi kombinasi operator yang mungkin
    string symbols = "+-*/";

    srand((unsigned int)time(NULL));
```

```
    opCombination = repeatingCombination(symbols, symbols.length(), 3);
}

vector<string> find(vector<int> nums)
{
    // mengembalikan semua solusi game 24 untuk nilai-nilai kartu tertentu

    set<vector<int>> permutation = permute(nums, nums.size());
    vector<string> res;

    if (!isConfigured)
    {
        config();
    }

    for (auto elmt : permutation)
    {
        for (auto ops : opCombination)
        {
            for (int i = 0; i < 5; i++)
            {
                if (float_abs(evaluator(elmt, ops, i) - 24) < FLT_EPSILON)
                {
                    res.push_back(generateString(elmt, ops, i));
                }
            }
        }
    }

    return res;
}
```

3.3 IO.cpp

```
using namespace std;
#include <iostream>
#include <fstream>
#include <istream>
#include <regex>

// from src
#include "./headers/IO.h"
#include "./headers/card.h"

string fixExtension(string fileName)
{
    // make sure file extension is txt

    int idx = fileName.find('.');

    if (idx == string::npos)
    {
        return fileName + ".txt";
    }

    return fileName.substr(0, idx) + ".txt";
}

string removeExtraSpace(string s)
{
    // menghilangkan extra space pada suatu string

    return regex_replace(s, regex("^ +| +$|( ) +"), "$1");
}

void printVector(vector<string> outputs)
{
```

```
    // mencetak vektor string

    int l = outputs.size();

    for(int i = 0; i < l-1; i++)
    {
        cout << outputs[i] + " ";
    }

    cout << outputs[l-1];
}

void printVector(vector<int> outputs)
{
    // mencetak vektor angka

    int l = outputs.size();

    for(int i = 0; i < l-1; i++)
    {
        cout << to_string(outputs[i]) + " ";
    }

    cout << outputs[l-1];
}

void nl()
{
    cout << "\n";
}


tuple<bool, vector<int>> isInputValid(string input)
{
    // memeriksa apakah input 4 simbol kartu valid

    bool isValid = false;
    vector<string> inputVector;
    vector<int> inputNums;

    if (!isCardString(input))
    {
        cout << "\nInput kartu hanya dapat berupa angka [2, 10] atau karakter A,
J, Q, dan K (huruf kapital)!\n";
        nl();
    }

    else{

        inputVector = split(input);
        inputNums = cardToNum(inputVector);

        if (isNumCardsValid(inputNums))
        {
            cout << "\nInput kartu berhasil!\n";
            nl();

            isValid = true;
        }
    }

    tuple<bool, vector<int>> res (isValid, inputNums);

    return res;
}

vector<int> getCards()
{
```

```
    // menerima input kartu melalui konsol hingga input valid

    bool stop = false;
    string input;
    vector<string> inputVector;
    tuple<bool, vector<int>> res {};

    while (!stop)
    {
        cout << "\nMasukkan 4 simbol kartu : ";
        getline(cin >> ws, input);

        res = isInputValid(removeExtraSpace(input));
        stop = get<0>(res);

    }

    return get<1>(res);
}

vector<int> getInputFile()
{
    // menerima input file berupa angka-angka kartu

    bool stop = false;
    string input, fileName;
    vector<string> inputVector;
    tuple<bool, vector<int>> res {};

    string dist = "../test/input/";

    while (!stop)
    {
        cout << "Masukkan nama file: " + dist;
        getline(cin >> ws, fileName);

        ifstream fileStream(dist + fixExtension(fileName));

        if (fileStream.is_open())
        {
            getline(fileStream, input);

            res = isInputValid(removeExtraSpace(input));
            stop = get<0>(res);

            fileStream.close();
        }

        else
        {
            cout << "\nGagal membuka file!\n[Pastikan nama file benar]\n";
            nl();
        }

    }

    return get<1>(res);
}

void printResult(vector<string> outputs, float time)
{
    int l = outputs.size();

    cout << "\nBanyak solusi ditemukan : " + to_string(l) + "\n";

    nl();

    for(int i = 0; i < l; i++)
```

```
    {
        cout << outputs[i] + "\n";
    }

    nl();

    cout << "Waktu eksekusi : " + to_string(time / 1000) + "ms\n";

    nl();
}

void writeFile(vector<string> outputs, float time)
{
    // write outputs to a file

    string dist = "../test/output/", fileName;

    cout << "Masukkan nama file: " + dist;
    getline(cin >> ws, fileName);

    ofstream fileStream(dist + fixExtension(fileName));

    if (fileStream.is_open())
    {
        int l = outputs.size();

        fileStream << "Banyak solusi ditemukan : " + to_string(l) + "\n\n";

        for(int i = 0; i < l; i++)
        {
            fileStream << outputs[i] + "\n";
        }

        fileStream << "\nWaktu eksekusi : " + to_string(time / 1000) + "ms\n";

        fileStream.close();

        cout << "\nBerhasil menulis hasil ke file!\n";

        nl();
    }

    else
    {
        cout << "Gagal membuat file!\n[Masukkan ulang nama file]\n";
    }

    nl();



}

void welcome()
{
    cout << "     __  ___    ___     __ __    ______           ____  ______
___     __ __ \n";
    cout << "   /  |/  /   /   |   / //_/   / ____/          /  _/ /_  __/
|__ \\  / // /\n";
    cout << "  / /|_/ /   / /| |  /   /    / __/             / /    / /
__/ / / // /_ \n";
    cout << " / /  / /   / ___ | / /| |   / /___           _/ /    / /
/ __/ /__  __/ \n";
    cout << "/_/  /_/   /_/  |_|/_/ |_|  /_____/          /___/   /_/
/____/   /_/    \n";

}
```

```
void welcomeMenu()
{
    nl();
    cout << "[1] Start\n";
    cout << "[2] Exit\n";
    nl();
}

void inputMenu()
{
    nl();
    cout << "==== PILIHAN MASUKAN ====\n";
    nl();
    cout << "[1] Kartu Acak\n";
    cout << "[2] Input Konsol\n";
    cout << "[3] Input File\n";
    cout << "[4] Exit\n";
    nl();
}

void outputMenu()
{
    nl();
    cout << "Simpan Hasil?\n";
    cout << "[1] Ya\n";
    cout << "[2] Tidak\n";
    nl();
}

int inputChoice(int lowerBound, int upperBound)
{
    int input;

    while (true)
    {
        cout << "Pilihan : ";
        cin >> input;

        nl();

        if (cin.fail())
        {
            cout << "Pastikan input pilihan berupa angka!\n";
            cin.clear();
            cin.sync();

            nl();
        }
        else if (input < lowerBound || input > upperBound)
        {
            cout << "Input pilihan hanya dapat berada pada rentang [" +
to_string(lowerBound) + "," + to_string(upperBound) + "]!\n";
            nl();
        }

        else {
            break;
        }
    }

    return input;
}
```

3.4 main.cpp

```
```

```
using namespace std;

#include <iostream>
#include <chrono>

// from src
#include "./headers/card.h"
#include "./headers/game.h"
#include "./headers/IO.h"

int main()
{
    int input;
    vector<int> nums;
    vector<string> res;

    config();

    welcome();
    welcomeMenu();

    input = inputChoice(1, 2);

    if (input == 1)
    {
        do {

            inputMenu();

            input = inputChoice(1, 4);

            if (input != 4)
            {
                if (input == 1)
                {
                    nums = generateRandom();

                    nl();
                    cout << "Kartu acak yang diperoleh : ";
                    printVector(numToCard(nums));
                    nl();
                }

                if (input == 2)
                {
                    nums = getCards();
                }

                else if (input == 3)
                {
                    nums = getInputFile();
                }

                auto start = chrono::high_resolution_clock::now();

                res = find(nums);

                auto end = chrono::high_resolution_clock::now();

                auto duration = chrono::duration_cast<chrono::microseconds>(end -
start);

                float time = duration.count();

                printResult(res, time);

                outputMenu();
```

```
                input = inputChoice(1, 2);

                if (input == 1)
                {
                    writeFile(res, time);
                }
            }

        } while (input != 4);
    }

    return 0;

}
```

BAB IV

Eksperimen

Berikut adalah hasil percobaan beberapa kasus uji terhadap program yang telah dibuat.

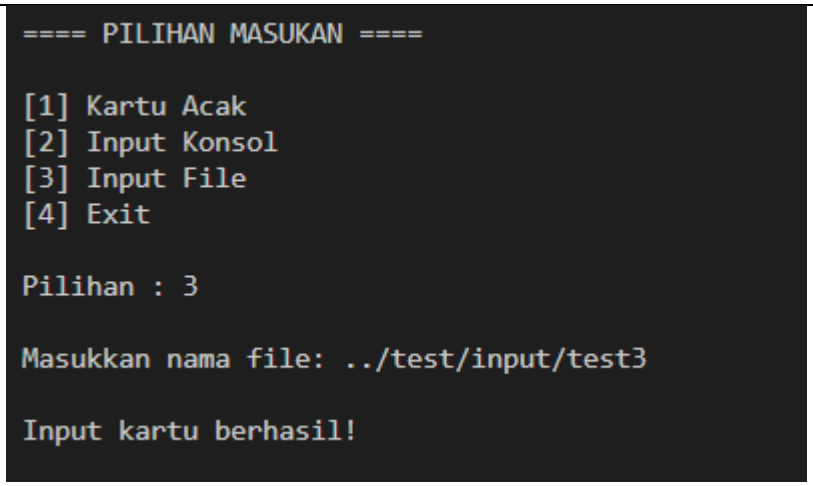
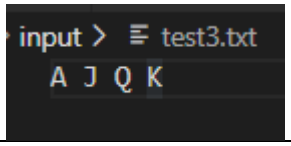
No.	*Input/ Output*	Gambar
1	*Input*	

	Output	Kartu acak yang diperoleh : 7 Q 6 K Banyak solusi ditemukan : 92 ((6 - 7) + Q) + K (6 - 7) + (Q + K) (6 - (7 - Q)) + K 6 - (7 - (Q + K)) 6 - ((7 - Q) - K) ((6 - 7) + K) + Q (6 - 7) + (K + Q) (6 - (7 - K)) + Q 6 - (7 - (K + Q)) 6 - ((7 - K) - Q) ((6 + Q) - 7) + K (6 + (Q - 7)) + K 6 + ((Q - 7) + K) (6 + Q) - (7 - K) 6 + (Q - (7 - K)) ((6 + Q) + K) - 7 (6 + Q) + (K - 7) (6 + (Q + K)) - 7 6 + ((Q + K) - 7) 6 + (Q + (K - 7)) ((6 + K) - 7) + Q (6 + (K - 7)) + Q 6 + ((K - 7) + Q) (6 + K) - (7 - Q) 6 + (K - (7 - Q)) (6 * (K - 7)) - Q ((6 + K) + Q) - 7 (6 + K) + (Q - 7) (6 + (K + Q)) - 7 6 + ((K + Q) - 7) 6 + (K + (Q - 7)) ((Q + 6) - 7) + K (Q + (6 - 7)) + K Q + ((6 - 7) + K) (Q + 6) - (7 - K) Q + (6 - (7 - K)) ((Q + 6) + K) - 7 (Q + 6) + (K - 7) (Q + (6 + K)) - 7 Q + ((6 + K) - 7) Q + (6 + (K - 7)) ((Q - 7) + 6) + K (Q - 7) + (6 + K) (Q - (7 - 6)) + K Q - (7 - (6 + K)) Q - ((7 - 6) - K)	

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

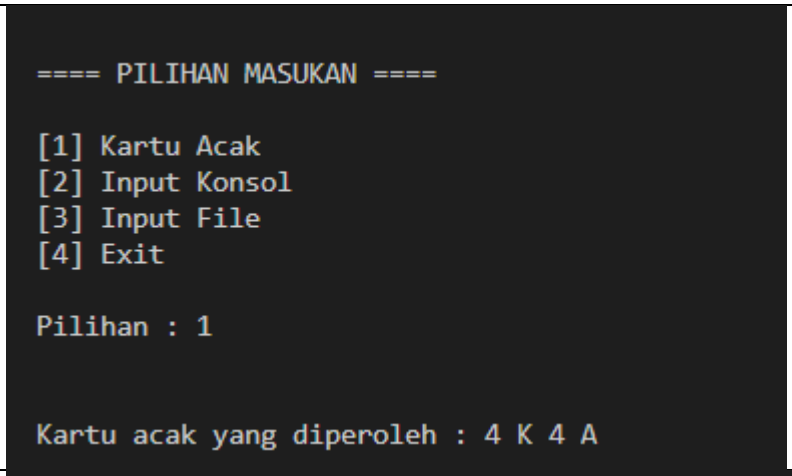
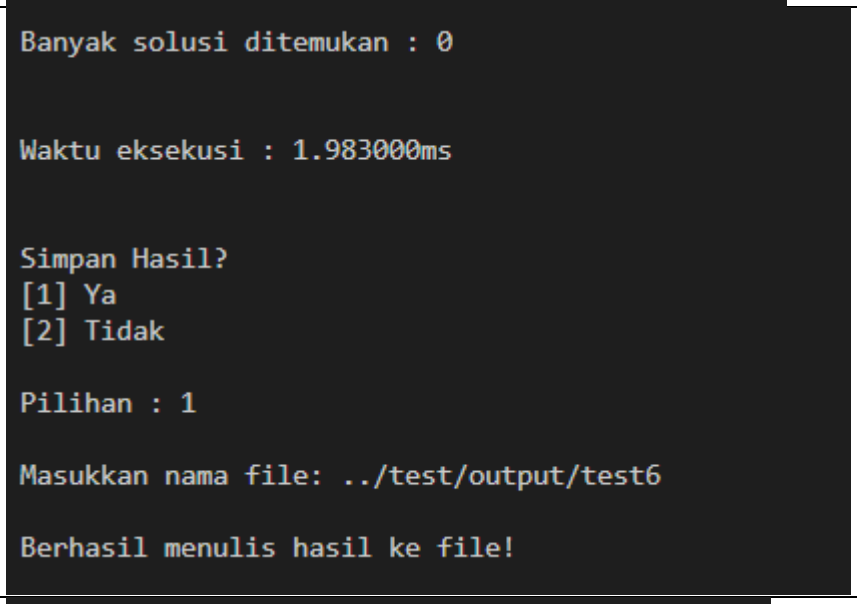
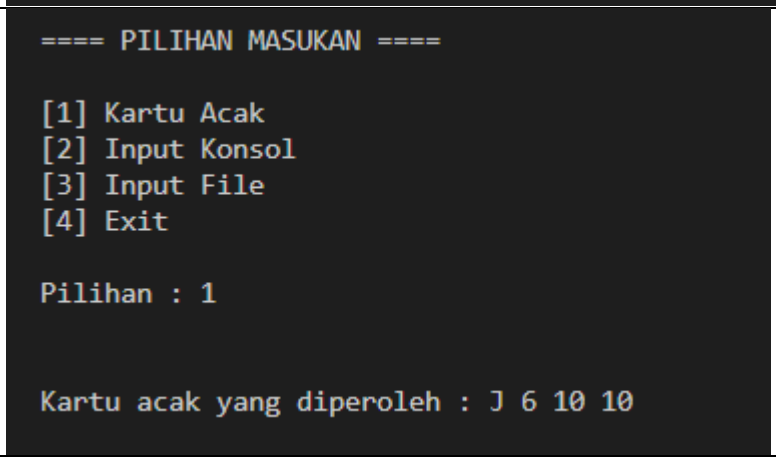
		K + ((Q - 7) + 6) (K + Q) - (7 - 6) K + (Q - (7 - 6)) Waktu eksekusi : 4.003000ms Simpan Hasil? [1] Ya [2] Tidak Pilihan : 1 Masukkan nama file: ../test/output/test1 Berhasil menulis hasil ke file!
2	*Input*	==== PILIHAN MASUKAN ==== [1] Kartu Acak [2] Input Konsol [3] Input File [4] Exit Pilihan : 2 Masukkan 4 simbol kartu : 1 2 3 4 Input kartu hanya dapat berupa angka [2, 10] atau karakter A, J, Q, dan K (huruf kapital)! Masukkan 4 simbol kartu : 2 3 4 5 Input kartu berhasil!

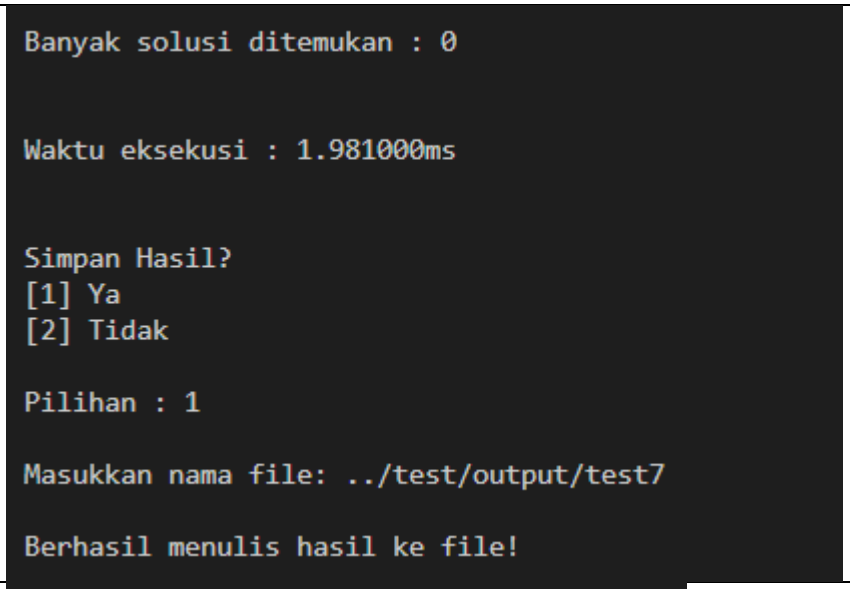
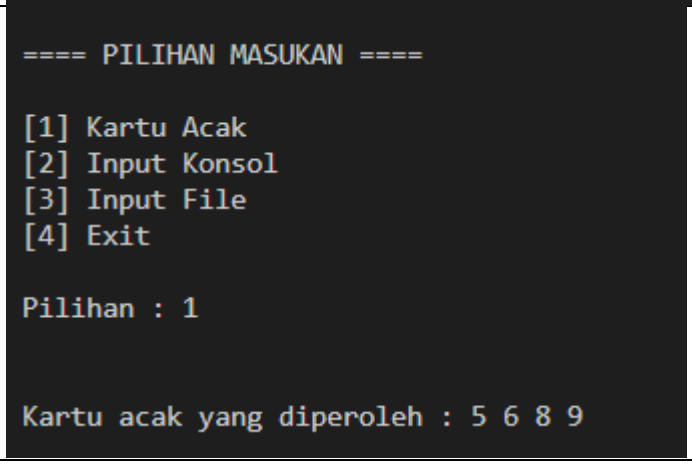
	Output	Banyak solusi ditemukan : 40 2 * ((3 + 4) + 5) 2 * (3 + (4 + 5)) 2 * ((3 + 5) + 4) 2 * (3 + (5 + 4)) 2 * ((4 + 3) + 5) 2 * (4 + (3 + 5)) 2 * ((4 + 5) + 3) 2 * (4 + (5 + 3)) 2 * ((5 + 3) + 4) 2 * (5 + (3 + 4)) 2 * ((5 + 4) + 3) 2 * (5 + (4 + 3)) ((3 - 2) + 5) * 4 (3 - (2 - 5)) * 4 ((3 + 4) + 5) * 2 (3 + (4 + 5)) * 2 ((3 + 5) - 2) * 4 (3 + (5 - 2)) * 4 ((3 + 5) + 4) * 2 (3 + (5 + 4)) * 2 4 * ((3 - 2) + 5) 4 * (3 - (2 - 5)) ((4 + 3) + 5) * 2 (4 + (3 + 5)) * 2 4 * ((3 + 5) - 2) 4 * (3 + (5 - 2)) 4 * ((5 - 2) + 3) 4 * (5 - (2 - 3)) ((4 + 5) + 3) * 2 (4 + (5 + 3)) * 2 4 * ((5 + 3) - 2) 4 * (5 + (3 - 2)) ((5 - 2) + 3) * 4 (5 - (2 - 3)) * 4 ((5 + 3) - 2) * 4 (5 + (3 - 2)) * 4 ((5 + 3) + 4) * 2 (5 + (3 + 4)) * 2 ((5 + 4) + 3) * 2 (5 + (4 + 3)) * 2 Waktu eksekusi : 2.999000ms

3	*Input*	 ==== PILIHAN MASUKAN ==== [1] Kartu Acak [2] Input Konsol [3] Input File [4] Exit Pilihan : 3 Masukkan nama file: ../test/input/test3 Input kartu berhasil!  input > test3.txt A J Q K

	Output	Banyak solusi ditemukan : 32 (A * Q) * (K - J) A * (Q * (K - J)) ((A * K) - J) * Q (A * (K - J)) * Q A * ((K - J) * Q) (Q * A) * (K - J) Q * ((A * K) - J) Q * (A * (K - J)) (Q / A) * (K - J) Q / (A / (K - J)) Q * (K - (A * J)) Q * ((K * A) - J) Q * ((K / A) - J) (Q * (K - J)) * A Q * ((K - J) * A) Q * (K - (J * A)) (Q * (K - J)) / A Q * ((K - J) / A) Q * (K - (J / A)) (K - (A * J)) * Q ((K * A) - J) * Q ((K / A) - J) * Q ((K - J) * A) * Q (K - J) * (A * Q) (K - (J * A)) * Q ((K - J) / A) * Q (K - (J / A)) * Q (K - J) / (A / Q) ((K - J) * Q) * A (K - J) * (Q * A) ((K - J) * Q) / A (K - J) * (Q / A) Waktu eksekusi : 4.008000ms
4	*Input*	==== PILIHAN MASUKAN ==== [1] Kartu Acak [2] Input Konsol [3] Input File [4] Exit Pilihan : 1 Kartu acak yang diperoleh : 6 2 7 7

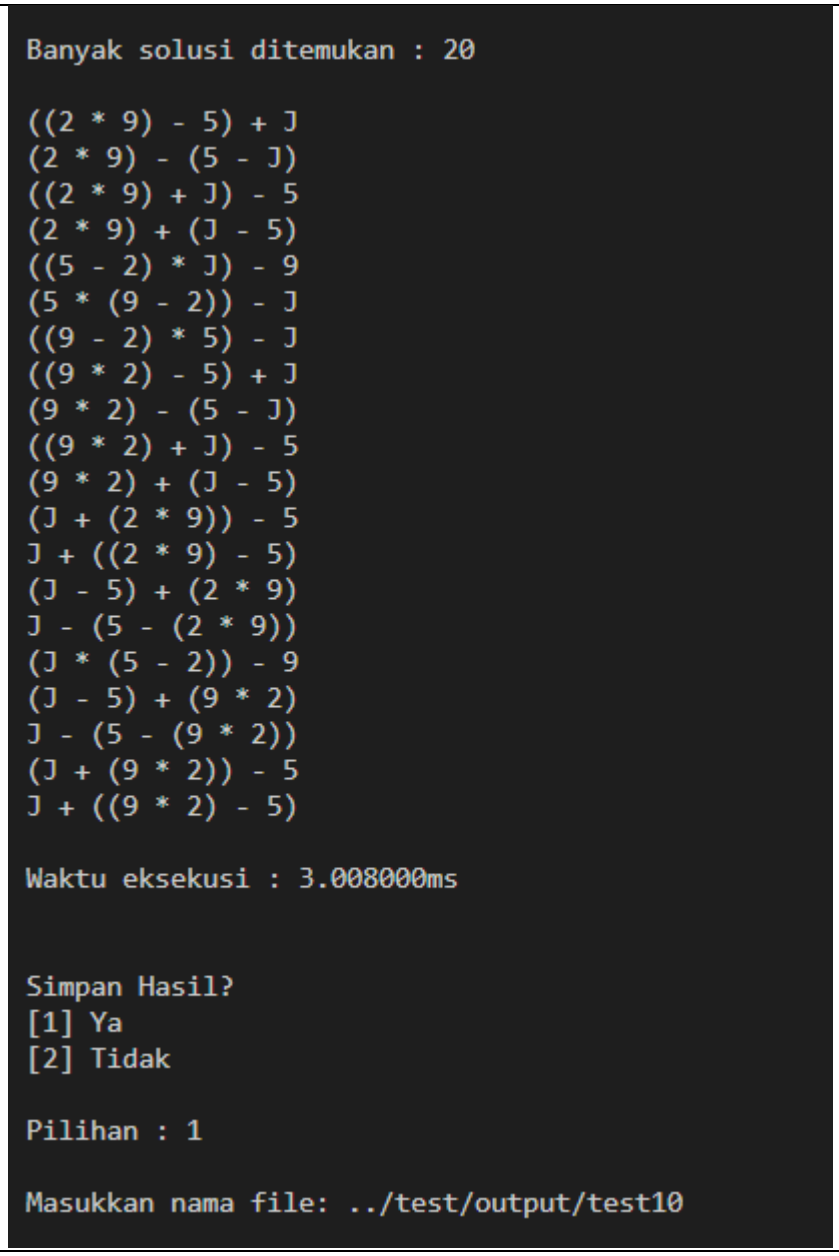
	Output	Banyak solusi ditemukan : 0 Waktu eksekusi : 1.989000ms Simpan Hasil? [1] Ya [2] Tidak Pilihan : 1 Masukkan nama file: ../test/output/test4 Berhasil menulis hasil ke file!
5	*Input*	==== PILIHAN MASUKAN ==== [1] Kartu Acak [2] Input Konsol [3] Input File [4] Exit Pilihan : 1 Kartu acak yang diperoleh : 6 10 8 J
	Output	Banyak solusi ditemukan : 2 (10 * (J - 8)) - 6 ((J - 8) * 10) - 6 Waktu eksekusi : 4.011000ms Simpan Hasil? [1] Ya [2] Tidak Pilihan : 1 Masukkan nama file: ../test/output/test5 Berhasil menulis hasil ke file!

6	*Input*	
	Output	
7	*Input*	

	Output	
8	*Input*	

	Output	Banyak solusi ditemukan : 32 ((5 + 8) - 9) * 6 (5 + (8 - 9)) * 6 ((5 - 9) + 8) * 6 (5 - (9 - 8)) * 6 6 * ((5 + 8) - 9) 6 * (5 + (8 - 9)) 6 * ((5 - 9) + 8) 6 * (5 - (9 - 8)) 6 * ((8 + 5) - 9) 6 * (8 + (5 - 9)) 6 * ((8 - 9) + 5) 6 * (8 - (9 - 5)) ((6 + 9) / 5) * 8 (6 + 9) / (5 / 8) ((6 + 9) * 8) / 5 (6 + 9) * (8 / 5) (8 / 5) * (6 + 9) 8 / (5 / (6 + 9)) ((8 + 5) - 9) * 6 (8 + (5 - 9)) * 6 (8 / 5) * (9 + 6) 8 / (5 / (9 + 6)) (8 * (6 + 9)) / 5 8 * ((6 + 9) / 5) ((8 - 9) + 5) * 6 (8 - (9 - 5)) * 6 (8 * (9 + 6)) / 5 8 * ((9 + 6) / 5) ((9 + 6) / 5) * 8 (9 + 6) / (5 / 8) ((9 + 6) * 8) / 5 (9 + 6) * (8 / 5) Waktu eksekusi : 2.986000ms
9	*Input*	==== PILIHAN MASUKAN ==== [1] Kartu Acak [2] Input Konsol [3] Input File [4] Exit Pilihan : 1 Kartu acak yang diperoleh : 9 9 6 K

	Output	Banyak solusi ditemukan : 0 Waktu eksekusi : 1.990000ms Simpan Hasil? [1] Ya [2] Tidak Pilihan : 1 Masukkan nama file: ../test/output/test9 Berhasil menulis hasil ke file!
10	*Input*	==== PILIHAN MASUKAN ==== [1] Kartu Acak [2] Input Konsol [3] Input File [4] Exit Pilihan : 1 Kartu acak yang diperoleh : 2 9 5 J

	Output	 Banyak solusi ditemukan : 20 ((2 * 9) - 5) + J (2 * 9) - (5 - J) ((2 * 9) + J) - 5 (2 * 9) + (J - 5) ((5 - 2) * J) - 9 (5 * (9 - 2)) - J ((9 - 2) * 5) - J ((9 * 2) - 5) + J (9 * 2) - (5 - J) ((9 * 2) + J) - 5 (9 * 2) + (J - 5) (J + (2 * 9)) - 5 J + ((2 * 9) - 5) (J - 5) + (2 * 9) J - (5 - (2 * 9)) (J * (5 - 2)) - 9 (J - 5) + (9 * 2) J - (5 - (9 * 2)) (J + (9 * 2)) - 5 J + ((9 * 2) - 5) Waktu eksekusi : 3.008000ms Simpan Hasil? [1] Ya [2] Tidak Pilihan : 1 Masukkan nama file: ../test/output/test10

Lampiran

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 Github : https://github.com/Enliven26/Tucil1_13521148