

**LAPORAN TUGAS KECIL I**  
**IF2211 STRATEGI ALGORITMA**  
**Penerapan Algoritma *Brute Force* Pada Permainan Kartu 24**

Dosen Pengampu : Dr. Ir. Rinaldi, M.T



**Disusun Oleh :**

Nama : Husnia Munzayana

NIM : 13521077

Kelas : K01

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**BANDUNG**  
**2022/2023**

## DAFTAR ISI

<b>Bab 1 Deskripsi Masalah</b>	1
<b>Bab 2 Teori Singkat</b>	2
2.1 Algoritma <i>Brute Force</i>	2
2.2 Kombinasi dan Permutasi	2
<b>Bab 3 Implementasi Algoritma</b>	4
3.1 Algoritma <i>Brute Force</i>	4
3.2 Source Program	5
3.2.1 <i>calculation.cpp</i>	5
3.2.1 <i>main.cpp</i>	7
3.3 Link Repositori	17
3.4 Check List Pengerjaan	17
<b>Bab 4 Percobaan Program</b>	18
4.1 Percobaan 1	18
4.2 Percobaan 2	18
4.3 Percobaan 3	18
4.4 Percobaan 4	19
4.5 Percobaan 5	19
4.6 Percobaan 6	19
4.7 Percobaan 7	20
4.8 Percobaan 8	20
<b>Daftar Pustaka</b>	21

# Bab 1

## Deskripsi Masalah

Permainan kartu 24 adalah permainan kartu aritmetika 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 ( $\times$ ), divisi (/) dan tanda kurung ( ( ) ). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas\*. Tugas Anda adalah merancang algoritma *brute force* untuk menemukan solusi dari permainan kartu 24 tersebut dengan spesifikasi sebagai berikut:

- a. Tulislah program sederhana dalam Bahasa C/C++/Java yang mengimplementasikan algoritma *Brute Force* untuk mencari seluruh solusi permainan kartu 24.
- b. Input 4 angka/huruf yang terdiri dari: (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K). 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.
- c. Output:
  1. Banyaknya solusi yang ditemukan.
  2. Solusi dari permainan kartu 24 ditampilkan di layar dan terdapat opsi untuk menyimpan solusi dalam file text.
  3. Waktu eksekusi program (tidak termasuk waktu pembacaan file input).

\* Dikutip dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf>

## Bab 2

### Teori Singkat

#### 2.1 Algoritma *Brute Force*

Algoritma *Brute Force* adalah pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan dengan sangat sederhana, langsung, dan jelas caranya. Contohnya dalam penyelesaian persoalan pencarian elemen terbesar/terkecil, *sequential search*, algoritma pengurutan (*selection sort* dan *bubble sort*), dan analisis pencocokan string. Penyelesaian persoalan dengan algoritma *Brute Force* umumnya menghasilkan solusi yang kurang efisien dan relatif lambat, walaupun terdapat beberapa permasalahan yang hanya dapat di selesaikan secara *Brute Force*. Hal ini disebabkan karena algoritma *Brute Force* mempertimbangkan untuk menggunakan cara yang sederhana dan implementasi yang mudah, dibandingkan harus memikirkan cara yang rumit yang lebih efisien. Akan tetapi, algoritma *brute force* umumnya dapat diterapkan pada hampir Sebagian besar permasalahan dan lebih mudah dimengerti karena algoritma pemecahan masalah yang digunakan relatif lebih sederhana.

*Exhaustive search* adalah salah satu metode pencarian dengan pendekatan *Brute Force* pada objek-objek kombinatoris. Pencarian dilakukan dengan men-generate atau mengenumerasi setiap kemungkinan untuk kemudian dievaluasi hingga menemukan elemen yang diinginkan. Algoritma ini dapat di optimasi dengan Teknik *Heuristic* untuk mengeliminasi kemungkinan solusi dengan pendekatan yang tidak formal, misal berdasarkan penilaian intuitif, terkaan, atau akal sehat.

#### 2.2 Kombinasi dan Permutasi

Kombinatorial adalah cabang matematika untuk menghitung (*counting*) jumlah penyusunan objek-objek tanpa harus mengenumerasi semua kemungkinan susunannya. Terdapat dua kaidah atau prinsip dasar perhitungan dalam kombinatorial :

a. Kaidah perkalian (*rule of product*)

Percobaan 1 :  $p$

Percobaan 2 :  $q$

Percobaan 1 **dan** Percobaan 2 :  $p \times q$

b. Kaidah penjumlahan (*rule if sum*)

Percobaan 1 :  $p$

Percobaan 2 :  $q$

Percobaan 1 **atau** Percobaan 2 :  $p + q$

Permutasi adalah bentuk khusus penerapan prinsip dasar perkalian. Permutasi sendiri berarti jumlah kemungkinan pengaturan objek-objek dengan memperhatikan urutan. Menurut kaidah perkalian, permutasi dari  $n$  objek adalah :

$$P(n, n) = n. (n - 1). (n - 2) \dots 2.1 = n!$$

Permutasi  $r$  dari  $n$  elemen :

$$P(n, r) = n. (n - 1). (n - 2) \dots (n - (r - 1)) = \frac{n!}{(n - r)!}$$

Permutasi  $r$  dari  $n$  elemen :

$$P(n, r) = n. (n - 1). (n - 2) \dots (n - (r - 1)) = \frac{n!}{(n - r)!}$$

Permutasi  $n$  elemen dengan  $n_1, n_2, \dots, n_k$  elemen yang sama :

$$P(n; n_1, n_2, n_3, \dots, n_k) = \frac{P(n, n)}{n_1! n_2! n_3! \dots n_k!} = \frac{n!}{n_1! n_2! n_3! \dots n_k!}$$

Permutasi sendiri memiliki bentuk khusus yang biasa disebut kombinasi. Berbeda dengan permutasi, kaidah kombinasi tidak memperhatikan urutan pengaturan objek-objek. Jumlah kombinasi  $r$  elemen dari  $n$  elemen dapat dicari dengan rumus berikut :

$$C(n, r) = \binom{n}{r} = \frac{n. (n - 1). (n - 2) \dots (n - (r - 1))}{r!} = \frac{n!}{r! (n - r)!}$$

## Bab 3

### Implementasi Algoritma

#### 3.1 Algoritma *Brute Force*

Persoalan permainan kartu 24 dapat dipecahkan dengan menerapkan algoritma *Brute Force* dengan teknik *Exhaustive Search* pada beberapa elemen. Penerapan algoritma *Brute Force* pada permainan kartu 24 dilakukan dengan langkah berikut:

1. Misalkan masukkan 4 kartu sebagai variabel  $x_1, x_2, x_3$ , dan  $x_4$ , sedangkan operasi aritmatika yang digunakan dimisalkan  $o$ .
2. Enumerasi setiap kemungkinan susunan 4 kartu, antara lain:

$$\begin{array}{cccc}
 \langle x_1, x_2, x_3, x_4 \rangle & \langle x_2, x_1, x_3, x_4 \rangle & \langle x_3, x_1, x_2, x_4 \rangle & \langle x_4, x_1, x_2, x_3 \rangle \\
 \langle x_1, x_2, x_4, x_3 \rangle & \langle x_2, x_1, x_4, x_3 \rangle & \langle x_3, x_1, x_4, x_2 \rangle & \langle x_4, x_1, x_3, x_2 \rangle \\
 \langle x_1, x_3, x_2, x_4 \rangle & \langle x_2, x_3, x_1, x_4 \rangle & \langle x_3, x_2, x_1, x_4 \rangle & \langle x_4, x_2, x_1, x_3 \rangle \\
 \langle x_1, x_3, x_4, x_2 \rangle & \langle x_2, x_3, x_4, x_1 \rangle & \langle x_3, x_2, x_4, x_1 \rangle & \langle x_4, x_2, x_3, x_1 \rangle \\
 \langle x_1, x_4, x_2, x_3 \rangle & \langle x_2, x_4, x_1, x_3 \rangle & \langle x_3, x_4, x_1, x_2 \rangle & \langle x_4, x_3, x_1, x_2 \rangle \\
 \langle x_1, x_4, x_3, x_2 \rangle & \langle x_2, x_4, x_3, x_1 \rangle & \langle x_3, x_4, x_2, x_1 \rangle & \langle x_4, x_3, x_2, x_1 \rangle
 \end{array}$$

atau dengan rumus permutasi didapatkan bahwa terdapat  $P(4,4) = 4! = 24$  kemungkinan susunan kartu atau angka.

3. Susunan di atas berlaku jika tidak ada elemen yang sama pada setiap variabel  $x_1, x_2, x_3$ , dan  $x_4$ . Untuk memastikan tidak ada susunan berulang, maka untuk setiap susunan variabel yang didapatkan akan dilakukan pengecekan pada list susunan yang telah dihasilkan sebelumnya. Jika susunan tersebut belum ada pada list, maka masukkan susunan tersebut sebagai elemen list.
4. Enumerasi setiap kemungkinan susunan penggunaan tanda kurung, antara lain :
  - a. Tanpa tanda kurung  

$$x_1 o x_2 o x_3 o x_4$$
  - b. Sepasang tanda kurung mengapit 2 angka atau variabel  

$$(x_1 o x_2) o x_3 o x_4 \quad x_1 o (x_2 o x_3) o x_4 \quad x_1 o x_2 o (x_3 o x_4)$$
  - c. Sepasang tanda kurung mengapit 3 angka atau variabel  

$$(x_1 o x_2 o x_3) o x_4 \quad x_1 o (x_2 o x_3 o x_4)$$
  - d. Dua pasang tanda kurung masing-masing mengapit 2 angka atau variabel  

$$(x_1 o x_2) o (x_3 o x_4)$$
  - e. Tanda kurung bersarang (*Nested*)  

$$((x_1 o x_2) o x_3) o x_4 \quad x_1 o ((x_2 o x_3) o x_4)$$

$$(x_1 o (x_2 o x_3)) o x_4 \quad x_1 o (x_2 o (x_3 o x_4))$$

Terdapat 11 kemungkinan susunan penggunaan tanda kurung. Walaupun dari beberapa kemungkinan susunan tanda kurung tersebut dapat menghasilkan hasil yang *similar*, dalam algoritma ini susunan tersebut tetap dianggap menjadi solusi yang berbeda.

5. Enumerasi setiap kemungkinan susunan operator. Terdapat 4 operator yang dapat digunakan dalam permainan kartu 24 (+, -, ×, ÷), sedangkan pada setiap operasi 4 kartu hanya memerlukan 3 operator. Sehingga susunan operator antara lain:

< + + + >	< + - + >	< + × - >	< + - - >
< + + - >	< + - × >	< + × × >	< + × + >
< + + ÷ >	< + - ÷ >	< + + × >	dan seterusnya ...

dengan kaidah perkalian didapatkan bahwa terdapat  $4.4.4 = 4^3 = 64$  kemungkinan susunan operator.

6. Kombinasikan setiap kemungkinan susunan kartu atau angka, susunan tanda kurung, serta susunan operator sehingga dihasilkan  $24 \cdot 11 \cdot 64 = 16896$  kemungkinan operasi aritmatika.
7. Evaluasi setiap kemungkinan operasi aritmetika yang telah didapatkan sehingga dapat dieliminasi operasi yang tidak menghasilkan nilai 24. Setiap evaluasi memiliki prioritas pengerjaannya masing-masing sehingga evaluasi operasi aritmetika tersebut dikategorikan menjadi 6 urutan pengerjaan:  
 Kategori A = operator 1 – operator 2 – operator 3  
 Kategori B = operator 1 – operator 3 – operator 2  
 Kategori C = operator 2 – operator 1 – operator 3  
 Kategori D = operator 2 – operator 3 – operator 1  
 Kategori E = operator 3 – operator 1 – operator 2  
 Kategori F = operator 3 – operator 2 – operator 1
8. Setelah dilakukan eliminasi, akan didapatkan *list* kemungkinan operasi 4 angka yang menghasilkan 24.

## 3.2 Source Program

### 3.2.1 *calculation.cpp*

```
#include <iostream>
#include <iomanip>

// Fungsi kalkulasi sesuai dengan urutan prioritas pengerjaan
float calculate(float input1, float input2, char op)
{
    float hasilbagi;
    switch (op)
    {
        case '+':
            return input1 + input2;
        case '-':
            return input1 - input2;
        case '*':
            return input1 * input2;
        case '/':
            if (input2 == 0)
                hasilbagi = -1;
            else
                hasilbagi = float(input1) / input2;
            return hasilbagi;
    }
}

float calculateA(float input[4], char op1, char op2, char op3)
```

```

{ // Skema perhitungan 123
    float res;
    res = calculate(input[0], input[1], op1);
    res = calculate(res, input[2], op2);
    res = calculate(res, input[3], op3);
    return res;
}
float calculateB(float input[4], char op1, char op2, char op3)
{ // Skema perhitungan 132
    float res, res1, res2;
    res1 = calculate(input[0], input[1], op1);
    res2 = calculate(input[2], input[3], op3);
    res = calculate(res1, res2, op2);
    return res;
}
float calculateC(float input[4], char op1, char op2, char op3)
{ // Skema perhitungan 213
    float res;
    res = calculate(input[1], input[2], op2);
    res = calculate(input[0], res, op1);
    res = calculate(res, input[3], op3);
    return res;
}
float calculateD(float input[4], char op1, char op2, char op3)
{ // Skema perhitungan 231
    float res;
    res = calculate(input[1], input[2], op2);
    res = calculate(res, input[3], op3);
    res = calculate(input[0], res, op1);
    return res;
}
float calculateE(float input[4], char op1, char op2, char op3)
{ // Skema perhitungan 312
    float res, res1, res2;
    res2 = calculate(input[2], input[3], op3);
    res1 = calculate(input[0], input[1], op1);
    res = calculate(res1, res2, op2);
    return res;
}
float calculateF(float input[4], char op1, char op2, char op3)
{ // Skema perhitungan 321
    float res;
    res = calculate(input[2], input[3], op3);
    res = calculate(input[1], res, op2);
    res = calculate(input[0], res, op1);
    return res;
}

```



### 3.2.1 main.cpp

```
// TUGAS KECIL 1 STRATEGI ALGORITMA
// Nama : Husnia Munzayana
// NIM : 13521077

#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>
#include <vector>
#include <ctime>
#include "calculation.cpp"
using namespace std;

// Inisiasi Variabel
string user[4], inp[4];
float inpNum[4];
int inpInteger[4];
char op[4] = {'+', '-', '*', '/'};
char op1, op2, op3, op4;
bool validInput = false, validSistem = false;
int count = 0;
float calResult;
vector<string> result;
vector<string> temp;
vector<vector<string>> permInput;
int inKartu;
char inputPil;
string namaFile;
clock_t start, endTime;

// Fungsi pengecekan apakah terdapat susunan temp sudah ada di
permInput
bool isThereVector(vector<string> temp, vector<vector<string>>
permInput)
{
    bool tempFound = true, found = false;
    int i = 0;
    while (i < permInput.size() and !found)
    {
        int j = 0;
        while (j < 4 and tempFound)
        {
            if (permInput[i][j] == temp[j])
            {
                if (j == 3)
                {
                    found = true;
                }
                j++;
            }
            else
            {
                tempFound = false;
            }
            i++;
        }
        return found;
    }
}

// Program Utama
```

```

int main()
{
    cout << "\n*** SELAMAT DATANG DI PERMAINAN KARTU 24 ***\n";
    cout << "\nPilih sistem pemilihan kartu:\n      (1) Input nilai
kartu sendiri\n      (2) Pilih kartu secara acak\nInput answer : ";
    while (!validSistem)
    { // Validasi masukkan user
        cin >> inKartu;
        if (inKartu == 1 or inKartu == 2)
            validSistem = true;
        else
            cout << "\nMasukkan tidak sesuai! Coba Ulangi (1/2)\nInput
answer : ";
    }

    if (inKartu == 2)
    { // Generate 4 kartu random
        string number[13] = {"A", "2", "3", "4", "5", "6", "7", "8",
"9", "10", "J", "Q", "K"};
        cout << "Kartu :";
        for (int i = 0; i < 4; i++)
        {
            user[i] = number[rand() % 13];
            cout << " " << user[i];
        }
        cout << "\n";
    }
    else
    { // Pengguna memasukkan sendiri 4 kartu yang dipilih
        cout << "\nMasukkan 4 nilai kartu ! (K/Q/J/A/2-10) \nInput
Kartu : ";
        while (!validInput)
        { // Validasi input
            cin >> user[0] >> user[1] >> user[2] >> user[3];
            validInput = true;
            int i = 0;
            while (i < 4 and validInput)
            {
                if (user[i] == "A" or user[i] == "2" or user[i] == "3"
or user[i] == "4" or user[i] == "5" or user[i] == "6" or user[i] ==
"7" or user[i] == "8" or user[i] == "9" or user[i] == "10" or user[i]
== "J" or user[i] == "Q" or user[i] == "K")
                    i++;
                else
                    validInput = false;
            }
            if (!validInput)
                cout << "\nMasukkan tidak sesuai! Coba Ulangi
(K/Q/J/A/2-10) \nInput Kartu : ";
        }
    }

    // Memulai perhitungan waktu eksekusi
    start = clock();
    // Permutasi susunan 4 angka/kartu
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            if (i != j)
            {

```

```

        for (int k = 0; k < 4; k++)
        {
            if (k != i and k != j)
            {
                temp.push_back(user[i]);
                temp.push_back(user[j]);
                temp.push_back(user[k]);
                temp.push_back(user[6 - i - j - k]);
                if (permInput.empty())
                    permInput.push_back(temp);
                else if (!isThereVector(temp, permInput))
                    permInput.push_back(temp);
                while (!temp.empty())
                    temp.pop_back();
            }
        }
    }
}

for (int l = 0; l < permInput.size(); l++)
{
    for (int m = 0; m < 4; m++)
    {
        inp[m] = permInput[l][m];
    }

    // Representasi nilai string kartu menjadi bilangan
    for (int i = 0; i < 4; i++)
    {
        if (inp[i] == "10")
            inpNum[i] = 10;
        else
        {
            switch (inp[i][0])
            {
                case 'A':
                    inpNum[i] = 1;
                    break;
                case 'J':
                    inpNum[i] = 11;
                    break;
                case 'Q':
                    inpNum[i] = 12;
                    break;
                case 'K':
                    inpNum[i] = 13;
                    break;
                default:
                    inpNum[i] = inp[i][0] - '0';
            }
        }
    }

    for (int i = 0; i < 4; i++)
    {
        inpInteger[i] = static_cast<int>(inpNum[i]);
    }

    // Kemungkinan susunan tanda kurung yang digunakan
    // Kemungkinan 1 : Tanpa tanda kurung

```

```

        for (int i = 0; i < 4; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                for (int k = 0; k < 4; k++)
                {
                    op1 = op[i];
                    op2 = op[j];
                    op3 = op[k];
                    if (op1 == '*' or op1 == '/')
                    {
                        if (op2 == '*' or op2 == '/')
                            calResult = calculateA(inpNum, op1, op2,
op3);
                        else
                        {
                            if (op3 == '*' or op3 == '/')
                                calResult = calculateB(inpNum, op1,
op2, op3);
                            else
                                calResult = calculateA(inpNum, op1,
op2, op3);
                        }
                    }
                    else
                    {
                        if (op2 == '*' or op2 == '/')
                        {
                            if (op3 == '*' or op3 == '/')
                                calResult = calculateD(inpNum, op1,
op2, op3);
                            else
                                calResult = calculateC(inpNum, op1,
op2, op3);
                        }
                        else
                        {
                            if (op3 == '*' or op3 == '/')
                                calResult = calculateE(inpNum, op1,
op2, op3);
                            else
                                calResult = calculateA(inpNum, op1,
op2, op3);
                        }
                    }
                }
            }
        }

        // Menyimpan susunan operasi yang menghasilkan
nilai 24
        if (calResult == 24)
        {
            count++;
            result.push_back(to_string(inpInteger[0]) + "
" + op1 + " " + to_string(inpInteger[1]) + " " + op2 + " " +
to_string(inpInteger[2]) + " " + op3 + " " +
to_string(inpInteger[3]));
        }
    }
}

// Kemungkinan 2 : Sepasang tanda kurung mengapit 2 angka

```

```

// (A o B) o C o D
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        for (int k = 0; k < 4; k++)
        {
            op1 = op[i];
            op2 = op[j];
            op3 = op[k];
            if (op2 == '*' or op2 == '/')
                calResult = calculateA(inpNum, op1, op2, op3);
            else
            {
                if (op3 == '*' or op3 == '/')
                    calResult = calculateB(inpNum, op1, op2,
op3);
                else
                    calResult = calculateA(inpNum, op1, op2,
op3);
            }

            // Menyimpan susunan operasi yang menghasilkan
nilai 24
            if (calResult == 24)
            {
                count++;
                result.push_back("(" +
to_string(inpInteger[0]) + " " + op1 + " " + to_string(inpInteger[1])
+ " " + op2 + " " + op3 +
to_string(inpInteger[2]) + " " + op3 +
to_string(inpInteger[3]));
            }
        }
    }
}

// A o (B o C) o D
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        for (int k = 0; k < 4; k++)
        {
            op1 = op[i];
            op2 = op[j];
            op3 = op[k];
            if (op1 == '*' or op1 == '/')
                calResult = calculateC(inpNum, op1, op2, op3);
            else
            {
                if (op3 == '*' or op3 == '/')
                    calResult = calculateD(inpNum, op1, op2,
op3);
                else
                    calResult = calculateC(inpNum, op1, op2,
op3);
            }

            // Menyimpan susunan operasi yang menghasilkan
nilai 24
            if (calResult == 24)

```

```

        {
            count++;
            result.push_back(to_string(inpInteger[0]) + "
" + op1 + " " + "(" + " " + to_string(inpInteger[1]) + " " + op2 + " "
+ to_string(inpInteger[2]) + " " + ")" + " " + op3 + " " +
to_string(inpInteger[3]));
        }
    }
}

// A o B o (C o D)
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        for (int k = 0; k < 4; k++)
        {
            op1 = op[i];
            op2 = op[j];
            op3 = op[k];
            if (op1 == '*' or op1 == '/')
                calResult = calculateE(inpNum, op1, op2, op3);
            else
            {
                if (op3 == '*' or op3 == '/')
                    calResult = calculateF(inpNum, op1, op2,
op3);
                else
                    calResult = calculateE(inpNum, op1, op2,
op3);
            }

            // Menyimpan susunan operasi yang menghasilkan
nilai 24
            if (calResult == 24)
            {
                count++;
                result.push_back(to_string(inpInteger[0]) + "
" + op1 + " " + to_string(inpInteger[1]) + " " + op2 + " " + "(" + " "
+ to_string(inpInteger[2]) + " " + op3 + " " +
to_string(inpInteger[3]) + " " + ")");
            }
        }
    }
}

// Kemungkinan 3
// (A o B o C) o D
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        for (int k = 0; k < 4; k++)
        {
            op1 = op[i];
            op2 = op[j];
            op3 = op[k];
            if (op1 == '*' or op1 == '/')
                calResult = calculateA(inpNum, op1, op2, op3);
            else

```

```

        {
            if (op2 == '*' or op2 == '/')
                calResult = calculateC(inpNum, op1, op2,
op3);
            else
                calResult = calculateA(inpNum, op1, op2,
op3);
        }

        // Menyimpan susunan operasi yang menghasilkan
nilai 24
        if (calResult == 24)
        {
            count++;
            result.push_back("(" +
to_string(inpInteger[0]) + " " + op1 + " " + to_string(inpInteger[1])
+ " " + op2 + " " + to_string(inpInteger[2]) + " " + ") " + " " + op3 +
" " + to_string(inpInteger[3]));
        }
    }

    // A o (B o C o D)
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            for (int k = 0; k < 4; k++)
            {
                op1 = op[i];
                op2 = op[j];
                op3 = op[k];
                if (op2 == '*' or op2 == '/')
                    calResult = calculateD(inpNum, op1, op2, op3);
                else
                {
                    if (op3 == '*' or op3 == '/')
                        calResult = calculateF(inpNum, op1, op2,
op3);
                    else
                        calResult = calculateD(inpNum, op1, op2,
op3);
                }

                // Menyimpan susunan operasi yang menghasilkan
nilai 24
                if (calResult == 24)
                {
                    count++;
                    result.push_back(to_string(inpInteger[0]) + "
" + op1 + " " + "(" + " " + to_string(inpInteger[1]) + " " + op2 + " "
+ to_string(inpInteger[2]) + " " + op3 + " " +
to_string(inpInteger[3]) + " " + ")");
                }
            }
        }
    }

    // Kemungkinan 4
    // (A o B) o (C o D)

```

```

for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        for (int k = 0; k < 4; k++)
        {
            op1 = op[i];
            op2 = op[j];
            op3 = op[k];
            calResult = calculateB(inpNum, op1, op2, op3);

            // Menyimpan susunan operasi yang menghasilkan
nilai 24
            if (calResult == 24)
            {
                count++;
                result.push_back("(" +
to_string(inpInteger[0]) + " " + op1 + " " + to_string(inpInteger[1])
+ " " + ")" + " " + op2 + " " + "(" + " " + to_string(inpInteger[2]) +
" " + op3 + " " + to_string(inpInteger[3]) + " " + ")");
            }
        }
    }

    // Kemungkinan 5
    // ((A o B) o C) o D
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            for (int k = 0; k < 4; k++)
            {
                op1 = op[i];
                op2 = op[j];
                op3 = op[k];
                calResult = calculateA(inpNum, op1, op2, op3);

                // Menyimpan susunan operasi yang menghasilkan
nilai 24
                if (calResult == 24)
                {
                    count++;
                    result.push_back("(" +
to_string(inpInteger[0]) + " " + op1 + " " + to_string(inpInteger[1])
+ " " + ")" + " " + op2 + " " + to_string(inpInteger[2]) + " " + ")" +
" " + op3 + " " + to_string(inpInteger[3]));
                }
            }
        }
    }

    // (A o (B o C)) o D
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            for (int k = 0; k < 4; k++)
            {
                op1 = op[i];
                op2 = op[j];

```



```

        op3 = op[k];
        calResult = calculateC(inpNum, op1, op2, op3);

        // Menyimpan susunan operasi yang menghasilkan
nilai 24
        if (calResult == 24)
        {
            count++;
            result.push_back("(" +
to_string(inpInteger[0]) + " " + op1 + " " + "(" + " " +
to_string(inpInteger[1]) + " " + op2 + " " + to_string(inpInteger[2])
+ " " + ")") + " " + op3 + " " + to_string(inpInteger[3]));
        }
    }
}

// A o ((B o C) o D)
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        for (int k = 0; k < 4; k++)
        {
            op1 = op[i];
            op2 = op[j];
            op3 = op[k];
            calResult = calculateD(inpNum, op1, op2, op3);

            // Menyimpan susunan operasi yang menghasilkan
nilai 24
            if (calResult == 24)
            {
                count++;
                result.push_back(to_string(inpInteger[0]) + "
" + op1 + " " + "(" + " " + to_string(inpInteger[1]) + " " + op2 + "
" + to_string(inpInteger[2]) + " " + ")") + " " + op3 + " " +
to_string(inpInteger[3]) + " " + ")");
            }
        }
    }
}

// A o (B o (C o D))
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        for (int k = 0; k < 4; k++)
        {
            op1 = op[i];
            op2 = op[j];
            op3 = op[k];
            calResult = calculateF(inpNum, op1, op2, op3);

            // Menyimpan susunan operasi yang menghasilkan
nilai 24
            if (calResult == 24)
            {
                count++;

```

```

        result.push_back(to_string(inpInteger[0]) + "
" + op1 + " " + "(" + " " + to_string(inpInteger[1]) + " " + op2 + " "
+ "(" + " " + to_string(inpInteger[2]) + " " + op3 + " " +
to_string(inpInteger[3]) + " " + ")"));
    }
    }
    }

    // Akhir eksekusi program, menghitung durasi waktu eksekusi
    endTime = clock();
    double duration = double(endTime - start) /
double(CLOCKS_PER_SEC);

    // Menampilkan hasil
    if (count != 0)
    {
        cout << "\n"
            << count << " Solution Found!\n";
        for (auto i = result.begin(); i != result.end(); ++i)
        {
            cout << *i << "\n";
        }
    }
    else
    {
        cout << "Tidak ada solusi\n";
    }
    cout << "Execution time : " << fixed << duration <<
setprecision(5) << " seconds.\n";

    // Menyimpan solusi pada file .txt
    validInput = false;
    cout << "\nApakah Anda ingin menyimpan hasil ? (Y/N)\nInput answer
: ";
    while (!validInput)
    {
        cin >> inputPil;
        if (inputPil == 'Y' or inputPil == 'y')
        {
            validInput = true;
            cout << "Masukkan nama file : ";
            cin >> namaFile;
            ofstream file;
            file.open("../test/" + namaFile + ".txt");
            file << "Kartu : " << user[0] << " " << user[1] << " " <<
user[2] << " " << user[3] << "\n";
            if (count != 0)
            {
                file << count << " Solution Found!\n";
                for (auto i = result.begin(); i != result.end(); ++i)
                {
                    file << *i << "\n";
                }
            }
            else
            {
                file << "No solution found\n";
            }
        }
    }
}

```

```

        file << "Execution time : " << fixed << duration <<
setprecision(5) << " seconds.\n";
        file.close();
        cout << "Berhasil menyimpan solusi di '../test/" +
namaFile + ".txt'\n";
    }
    else if (inputPil == 'N' or inputPil == 'n')
        validInput = true;
    else
    {
        cout << "\nMasukkan tidak sesuai! Coba Ulangi (Y/N)\nInput
answer : ";
    }
}

cout << "\nPermainan berakhir. Terimakasih :) \n";
return 0;
}

```

### 3.3 Link Repositori

[https://github.com/munzayanahusn/Tucil1\\_13521077.git](https://github.com/munzayanahusn/Tucil1_13521077.git)

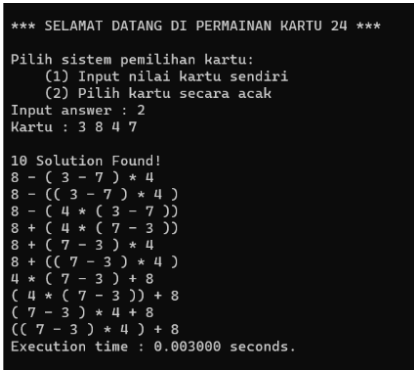
### 3.4 Check List Pengerjaan

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	✓	

## Percobaan Program

## 4.1 Percobaan 1

```
Input      : Pilih kartu acak (dihasilkan kartu 3 8 4 7)
Output     : 10 Solution Found!
```



```
*** SELAMAT DATANG DI PERMAINAN KARTU 24 ***

Pilih sistem pemilihan kartu:
(1) Input nilai kartu sendiri
(2) Pilih kartu secara acak
Input answer : 2
Kartu : 3 8 4 7

10 Solution Found!
8 - ( 3 - 7 ) * 4
8 - (( 3 - 7 ) * 4 )
8 - ( 4 * ( 3 - 7 ) )
8 + ( 4 * ( 7 - 3 ) )
8 + ( 7 - 3 ) * 4
8 + (( 7 - 3 ) * 4 )
4 * ( 7 - 3 ) + 8
( 4 * ( 7 - 3 ) ) + 8
( 7 - 3 ) * 4 + 8
(( 7 - 3 ) * 4 ) + 8
Execution time : 0.003000 seconds.
```

## 4.2 Percobaan 2

```

Input      : 6 7 3 10
Output     : 48 Solution Found!

```

### 4.3 Percobaan 3

```
Input      : 6 6 6 6
Output     : 17 Solution Found!

Masukkan 4 nilai kartu ! (K/Q/J/A/2-10)
Input Kartu : 6 6 6 6

17 Solution Found!
6 + 6 + 6 + 6
6 * 6 - 6 - 6
( 6 + 6 ) + 6 + 6
( 6 * 6 ) - 6 - 6
6 + ( 6 + 6 ) + 6
6 + 6 + ( 6 + 6 )
6 * 6 - ( 6 + 6 )
( 6 + 6 + 6 ) + 6
( 6 * 6 - 6 ) - 6
6 + ( 6 + 6 + 6 )
( 6 + 6 ) + ( 6 + 6 )
( 6 * 6 ) - ( 6 + 6 )
(( 6 + 6 ) + 6 ) + 6
(( 6 * 6 ) - 6 ) - 6
( 6 + ( 6 + 6 ) ) + 6
6 + (( 6 + 6 ) + 6 )
6 + ( 6 + ( 6 + 6 ) )
Execution time : 0.000000 seconds.
```



## 4.7 Percobaan 7

Input : K Q J A

Output : 54 Solution Found!

```
Masukkan 4 nilai kartu ! (K/Q/J/A/2-10)
Input Kartu : K Q J A
```

54 Solution Found!

```
( 13 - 11 ) * 12 * 1
( 13 - 11 ) * 12 / 1
( 13 - 11 ) * ( 12 * 1 )
( 13 - 11 ) * ( 12 / 1 )
(( 13 - 11 ) * 12 ) * 1
(( 13 - 11 ) * 12 ) / 1
( 13 - 11 ) * 1 * 12
( 13 - 11 ) / 1 * 12
( 13 - 11 + 1 ) * 12
( 13 - 11 / 1 ) * 12
( 13 - 11 ) * ( 1 * 12 )
( 13 - 11 ) / ( 1 / 12 )
(( 13 - 11 ) * 1 ) * 12
(( 13 - 11 ) / 1 ) * 12
( 13 - ( 11 * 1 ) ) * 12
( 13 - ( 11 / 1 ) ) * 12
( 13 - 1 * 11 ) * 12
( 13 + 1 - 11 ) * 12
( 13 / 1 - 11 ) * 12
(( 13 * 1 ) - 11 ) * 12
(( 13 / 1 ) - 11 ) * 12
( 13 - ( 1 * 11 ) ) * 12
12 * ( 13 - 11 ) * 1
12 * ( 13 - 11 ) / 1
12 * ( 13 - 11 * 1 )
12 * ( 13 - 11 / 1 )
```

```
( 12 * ( 13 - 11 ) ) * 1
( 12 * ( 13 - 11 ) ) / 1
12 * (( 13 - 11 ) * 1 )
12 * (( 13 - 11 ) / 1 )
12 * ( 13 - ( 11 * 1 ) )
12 * ( 13 - ( 11 / 1 ) )
12 * ( 13 - 1 * 11 )
12 * ( 13 * 1 - 11 )
12 * ( 13 / 1 - 11 )
12 * (( 13 * 1 ) - 11 )
12 * (( 13 / 1 ) - 11 )
12 * ( 13 - ( 1 * 11 ) )
12 * 1 * ( 13 - 11 )
12 / 1 * ( 13 - 11 )
12 * ( 1 * 13 - 11 )
( 12 * 1 ) * ( 13 - 11 )
( 12 / 1 ) * ( 13 - 11 )
12 * (( 1 * 13 ) - 11 )
12 * ( 1 * ( 13 - 11 ) )
12 / ( 1 / ( 13 - 11 ) )
1 * ( 13 - 11 ) * 12
( 1 * 13 - 11 ) * 12
(( 1 * 13 ) - 11 ) * 12
( 1 * ( 13 - 11 ) ) * 12
1 * (( 13 - 11 ) * 12 )
1 * 12 * ( 13 - 11 )
( 1 + 12 ) * ( 13 - 11 )
1 * ( 12 * ( 13 - 11 ) )
Execution time : 0.000000 seconds.
```

## 4.8 Percobaan 8

Input : J 10 7 7

Output : Tidak ada solusi

```
*** SELAMAT DATANG DI PERMAINAN KARTU 24 ***
```

Pilih sistem pemilihan kartu:

(1) Input nilai kartu sendiri

(2) Pilih kartu secara acak

Input answer : 1

Masukkan 4 nilai kartu ! (K/Q/J/A/2-10)

Input Kartu : J 10 7 7

Tidak ada solusi

Execution time : 0.000000 seconds.

## Daftar Pustaka

- Admin. (n.d.). *Exhaustive Search*. BrainKart. Diakses pada 22 Januari 2023 pukul 20.45 dari sumber [https://www.brainkart.com/article/Exhaustive-Search\\_8013/](https://www.brainkart.com/article/Exhaustive-Search_8013/)
- Admin. (n.d.). *Permutation and Combination - Definition, Formulas, Derivation, Examples*. Cuemath. Diakses pada 22 Januari 2023 pukul 20.34 dari sumber <https://www.cuemath.com/data/permutations-and-combinations/>
- Admin. (2020, June 26). *Rekomendasi Jadwal Rapat Menggunakan Algoritma Exhaustive Search*. Kita Informatika. Diakses pada 22 Januari 2023 pukul 21.10 dari sumber <http://www.kitainformatika.com/2020/04/rekomendasi-jadwal-rapat-menggunakan.html>
- Munir, Rinaldi (2020). *Bahan Kuliah IF2120 Matematika Diskrit – Kombinatorial (Bagian 1)*. Diakses pada 22 Januari 2023 pukul 20.15 dari sumber <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Kombinatorial-2020-Bagian1.pdf>
- Munir, Rinaldi (2020). *Bahan Kuliah IF2120 Matematika Diskrit – Kombinatorial (Bagian 2)*. Diakses pada 22 Januari 2023 pukul 20.18 dari sumber <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Kombinatorial-2020-Bagian2.pdf>
- Munir, Rinaldi (2022). *Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Brute Force (Bagian 1)*. Diakses pada 21 Januari 2023 pukul 21.10 dari sumber [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf)
- Munir, Rinaldi (2022). *Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Brute Force (Bagian 2)*. Diakses pada 21 Januari 2023 pukul 21.15 dari sumber [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf)