

TUGAS KECIL 01
PENYELESAIAN PERMAINAN KARTU 24
DENGAN ALGORITMA BRUTE FORCE
IF2211 – STRATEGI ALGORITMA



Disusun oleh:

13521066 Muhammad Fadhil Amri

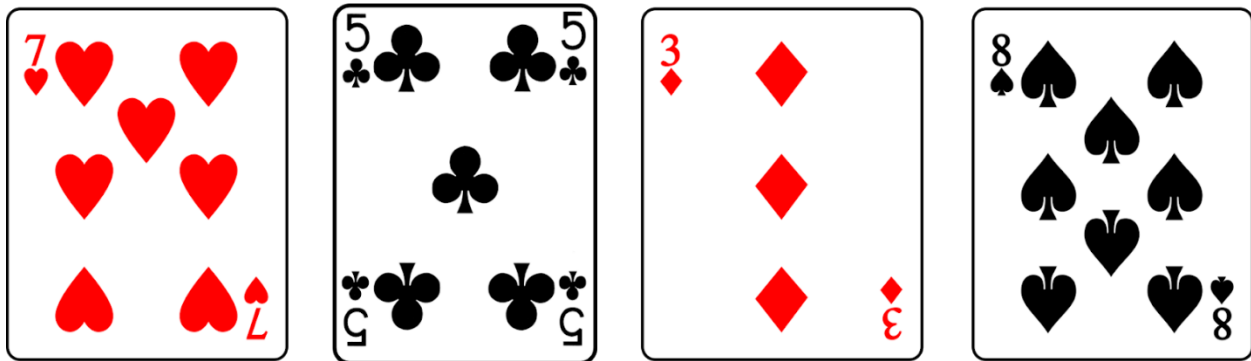
Institut Teknologi Bandung
Jl. Ganesha No. 10, Bandung 40132
2023

DAFTAR ISI

DAFTAR ISI.....	2
BAB I : DESKRIPSI MASALAH.....	3
BAB II: Algoritma Brute Force	4
BAB III : SOURCE PROGRAM.....	6
BAB IV : Eksperimen.....	16
A. Test Case 1 (2,5,A,K).....	16
B. TestCase2 (2 2 4 J)	17
C. Test Case 3 (6, 6, 6, 6).....	18
D. Test Case 4 (A, A, A, A)	18
E. Test Case 5 Random(4, 10, 3, Q)	19
F. Test Case 6 Random(10, 7, J, J)	20
G. Test Case 7 Random(10, 7, J, J).....	20
H. Test Case 7 Random(Q, J, 6, A).....	21
BAB V : <i>LINK REPOSITORY</i>	22
BAB VI : CHECKING TABLE	22

BAB I : DESKRIPSI MASALAH

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Permainan 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. (Paragraf di atas dikutip dari sini: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah2016/MakalahStima-2016-038.pdf>).



MAKE IT 24

BAB II: Algoritma Brute Force

A. Pengertian Algoritma Brute Force

Brute Force adalah sebuah pendekatan yang langsung (*straightforward*) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (*obvious way*).

B. Kelebihan Algoritma Brute Force

Berikut beberapa kelebihan yang dimiliki oleh algoritma brute force.

- 1) Algoritma brute force dapat digunakan untuk memecahkan hampir sebagian besar masalah.
- 2) Sederhana dan mudah dimengerti.
- 3) Menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, dan perkalian matriks.
- 4) Menghasilkan algoritma baku (*standard*) untuk tugas-tugas komputasi seperti penjumlahan atau perkalian N buah bilangan dan penentuan elemen minimum atau maksimum di tabel.

C. Kekurangan Algoritma Brute Force

Berikut beberapa kekurangan yang dimiliki oleh algoritma brute force.

- 1) Jarang menghasilkan algoritma yang efektif.
- 2) Lambat sehingga tidak dapat diterima
- 3) Tidak kreatif Teknik pemecahan masalah lainnya

D. Penerapan Algoritma Brute Force

Berikut Algoritma Brute Force yang diterapkan pada penyelesaian permainan kartu 24 (Solusi yang similar tetapi dengan bentuk yang berbeda dianggap sebagai solusi yang berbeda).

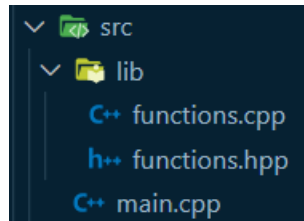
1. Lakukan input kartu yang akan digunakan hingga valid, yaitu empat buah kartu dan keempatnya merupakan kartu yang terdapat pada list (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King).
2. Lakukan permutasi pada susunan kartu yang digunakan untuk permainan. Hal ini ditujukan untuk mencari semua kemungkinan susunan kartu pada solusi. Karena ada empat buah kartu, jumlah maksimal susunan kartu yang mungkin adalah 24 susunan, yaitu saat keempat kartu yang terpilih berbeda. Sementara itu, jumlah minimal susunan kartu yang mungkin adalah satu susunan, yaitu saat keempat kartu yang terpilih sama.

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

3. Lakukan permutasi pada susunan operator yang akan digunakan. Pada permainan ini terdapat empat jenis operator yang digunakan, yaitu operator perkalian(*), pembagian(/), penjumlahan(+), dan pengurangan(-). Banyaknya susunan operator yang mungkin adalah 64 susunan karena terdapat tiga buah operator yang digunakan pada permainan.
4. Lakukan pembagian kasus penempatan tanda kurung () pada permainan. Singkatnya, pada permainan ini ada lima kemungkinan penempatan tanda kurung () yang efektif, yaitu.
 - a. (val1 op1 val2) op2 (val3 op3 val4)
 - b. ((val1 op1 val2) op2 val3) op3 val4
 - c. (val1 op1 (val2 op2 val3)) op3 val4
 - d. val1 op1 ((val2 op2 val3) op3 val4)
 - e. val1 op1 (val2 op2 (val3 op3 val4))
5. Evaluasi ekspresi untuk semua kombinasi susunan kartu, operator dan tanda kurung yang mungkin. Jika hasil evaluasi ekspresi sama dengan 24, ekspresi tersebut dimasukkan ke dalam kumpulan solusi persoalan. Pada program algoritma brute force ini dilakukan sedikit modifikasi yaitu, nilai absolut dari selisih hasil evaluasi dengan 24 adalah lebih kecil dari epsilon (pada kasus ini, $\epsilon = 10^{-12}$). Modifikasi ini ditujukan untuk normalisasi hasil operasi pecahan yang seharusnya menghasilkan bilangan bulat, tetapi karena format *double*, terdapat sedikit galat sehingga bilangan bulat tidak bisa dihasilkan.
6. Tampilkan banyak solusi pada persoalan dan semua bentuk solusi yang mungkin.

BAB III : SOURCE PROGRAM

Program dibuat dengan bahasa pemrograman C++. Program terdiri atas dua komponen utama, yaitu *main program* dan *helper program*. *Main program* terdapat pada file `main.cpp`, sedangkan *helper program* terdapat pada *folder lib* pada file `functions.cpp`.



Gambar 1. Susunan Program pada Folder `src`

A. `main.cpp`

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <ctime>
#include "lib/functions.hpp"

using namespace std;

// ALGORITMA
int main(){
    // Setup
    int input[4];
    clock_t time_start, time_end;
    vector<string> solutions;

    // START
    cout << endl<< "==== MAKE IT 24 =====" << endl << endl;

    getInput(input);
    time_start = clock();
    getSolution(input,&solutions);
    time_end = clock();
    double duration = (time_end-time_start)/((double)CLOCKS_PER_SEC);

    // CLOSING
    if (solutions.size()>0){ // Jika terdapat solusi
        // Menampilkan banyak solusi dan kumpulan solusinya
        char isSave;
        cout<<"Banyak Solusi: "<<solutions.size()<<endl;
        cout<<"Solusi: "<<endl;
        printSolution(solutions);
        cout<< endl;
    }
```

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
// Validasi opsi save solution
do{
    cout<<"Apakah ingin menyimpan solusi? (y/n): "<< endl;
    cin>>isSave;
} while (isSave!='y' && isSave!='n');

// Jika user memilih save file
if (isSave=='y'){
    string namaFile;
    cout<<"Masukkan Nama File Solusi: "<< endl;
    cin>>namaFile;
    saveFile(namaFile,solutions);
}
} else { // Jika tidak terdapat solusi
    cout<<"Tidak Ada Solusi"<<endl;
}
cout<<endl<<"Execution Time: "<<duration<<" s"<<endl;

return 0;
}
```

B. functions.hpp

```
#ifndef FUNCTIONS_HPP
#define FUNCTIONS_HPP
#include <string>
#include <cstdlib>
#include <vector>
#include <cmath>
using namespace std;

/**
 * @brief Tipe data untuk suatu operasi
 * @param res: Hasil
 * @param op: symbol operator
 */
You, yesterday | 1 author (You)
typedef struct{
    double res;
    char op;
} Operasi;

/**
 * @brief Tipe data untuk arrangement kartu
 * @param card1: Kartu posisi pertama
 * @param card2: Kartu posisi kedua
 * @param card3: Kartu posisi ketiga
 * @param card4: Kartu posisi keempat
 */
You, yesterday | 1 author (You)
typedef struct {
    int card1;
    int card2;
    int card3;
    int card4;
} Cards;
```

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
/**
 * @brief Tipe data untuk menyimpan address fungsi operator
 */
typedef Operasi (*functionPtr) (double, double);

// FUNGSI / PROSEDUR
/**
 * @brief Get the Input object
 * @brief Input bisa diberikan oleh user atau auto-generated
 * @return int*
 */
void getInput(int input[]);

/**
 * @brief Get the Solution object
 * @brief Mencari semua solusi yang mungkin
 * @param input: array of integer
 * @return string*
 */
void getSolution(int input[], vector<string> *output);

/**
 * @brief Mencari kombinasi tanda kurung yang memenuhi
 *
 * @param arrangement: Susunan Kartu
 * @param funcs: Kombinasi operator
 * @param output
 */
void solveParentheses(Cards arrangement, functionPtr funcs[], vector<string> *output);

/**
 * @brief Mencetak Solusi game 24
 * @param solution: Vektor Solusi
 */
void printSolution(vector<string> solution);
```

```
/**
 * @brief Menyimpan file solusi
 * @param namaFile
 * @param solutions
 */
void saveFile(string namaFile, vector<string> solutions);

/**
 * @brief Mengubah Kartu dari representasi string menjadi representasi integer
 * @param inputString
 * @param inputInt
 */
void cardStringToInt(string inputString[], int inputInt[]);

/**
 * @brief Mengubah kartu dari representasi integer menjadi string
 * @param inputInt
 * @param inputString
 */
void cardIntToString(int inputInt[], string inputString[]);

/**
 * @brief Menegcek apakah input(manual) valid
 * @param inputLines: baris input
 * @return true jika valid
 * @return false jika tidak valid
 */
bool isValidInput(string inputLines);

/**
 * @brief Mengecek apakah kartu termasuk kartu yang valid
 * @param stringCard: Kartu dengan representasi string
 * @return true jika valid
 * @return false jika tidak valid
 */
bool isValidCard(string stringCard);
```


Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
/**
 * @brief Mengecek apakah kartu sudah terdapat di dalam list
 * @param tempCard
 * @param list
 * @return true jika sudah terdapat
 * @return false jika tidak terdapat
 */
bool isExistCards(Cards tempCard, vector<Cards> list);

/**
 * @brief Menegcek apakah dua buah kartu sama
 * @param cards1
 * @param cards2
 * @return true jika sama
 * @return false jika berbeda
 */
bool isSameCards(Cards cards1, Cards cards2);

/**
 * @brief Operasi penjumlahan
 * @param a
 * @param b
 * @return Operasi
 */
Operasi tambah(double a, double b);

/**
 * @brief Operasi pengurangan
 * @param a
 * @param b
 * @return Operasi
 */
Operasi kurang(double a, double b);

/**
 * @brief Operasi perkalian
 * @param a
 * @param b
 * @return Operasi
 */
Operasi kali(double a, double b);
```

```
/**
 * @brief Operasi pembagian
 * @param a
 * @param b
 * @return Operasi
 */
Operasi bagi(double a, double b);
#endif
```

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

C. functions.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <ctime>
#include "functions.hpp"
```

```
Operasi tambah(double a, double b)
```

```
{
    // KAMUS LOKAL
    Operasi ops;
    // ALGORITMA
    ops.op = '+';
    ops.res = a+b;
    return ops;
}
```

```
Operasi kurang(double a, double b)
```

```
{
    // KAMUS LOKAL
    Operasi ops;
    // ALGORITMA
    ops.op = '-';
    ops.res = a-b;
    return ops;
}
```

```
Operasi kali(double a, double b)
```

```
{
    // KAMUS LOKAL
    Operasi ops;
    // ALGORITMA
    ops.op = '*';
    ops.res = a*b;
    return ops;
}
```

```
Operasi bagi(double a, double b)
```

```
{
    // KAMUS LOKAL
    Operasi ops;
    // ALGORITMA
    ops.op = '/';
    if (b != 0){
        ops.res = a/b;
    } else {
        ops.res = 9999; //undefine
    }
    return ops;
}
```

```
void cardStringToInt(string inputString[], int inputInt[])
```

```
{
    // KAMUS LOKAL
    int i;
    // ALGORITMA
    for(i=0; i<4; i++){
        if (inputString[i] == "A"){
            inputInt[i] = 1;
        } else if (inputString[i] == "J")
        {
            inputInt[i] = 11;
        } else if (inputString[i] == "Q")
        {
            inputInt[i] = 12;
        } else if (inputString[i] == "K")
        {
            inputInt[i] = 13;
        } else {
            inputInt[i] = stoi(inputString[i]);
        }
    }
}
```

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
void cardIntToString(int inputInt[], string inputString[])
{
    // KAMUS LOKAL
    int i;
    // ALGORITMA
    for(i=0; i<4; i++){
        if (inputInt[i] == 1){
            inputString[i] = "A";
        } else if (inputInt[i] == 11){
            inputString[i] = "J";
        } else if (inputInt[i] == 12){
            inputString[i] = "Q";
        } else if (inputInt[i] == 13){
            inputString[i] = "K";
        } else {
            inputString[i] = to_string(inputInt[i]);
        }
    }
}

// REALISASI FUNGSI/PROSEDUR
void getInput(int input[]){
    // KAMUS LOKAL
    int inputType, i;
    string inputString[4];
    // ALGORITMA
    // Validasi inputType
    do{
        cout<<"Tipe Input"<< endl;
        cout<<"1. Manual"<< endl;
        cout<<"2. Auto-Generated"<< endl;
        cout<<"Masukkan Jenis Input yang digunakan(1/2): ";
        cin>>inputType;
    } while (inputType!=1 && inputType!=2);
    // inputType sudah Valid
    cout<<endl;
```

Tugas Kecil 01 IF 2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
if (inputType == 1){ // Manual
    // setup
    string inputLines, arg;
    bool firstTry = true;
    bool validInput = false;

    // Validasi input
    do {
        cout<< "Masukkan 4 Kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dengan dipisahkan oleh satu spasi:"<<endl;
        if (firstTry){
            cin.ignore(); // Agar mencegah bug pada getline
        }
        getline(cin,inputLines);
        if(!isValidInput(inputLines)){
            cout<<"Masukan Tidak Sesuai"<< endl;
            firstTry = false;
        } else {
            validInput = true;
        }
    } while (!validInput);

    // Valid, collect the card
    stringstream semiArgs(inputLines);
    i=0;
    while (getline(semiArgs, arg, ' ')) {
        inputString[i] = arg;
        i++;
    }

    // Convert to int
    cardStringToInt(inputString, input);
} else { // Auto-Generated
    srand ( time(NULL) );
    input[0] = (rand() % 13) + 1;
    input[1] = (rand() % 13) + 1;
    input[2] = (rand() % 13) + 1;
    input[3] = (rand() % 13) + 1;
    // Convert dalam bentuk string
    cardIntToString(input,inputString);
}
```

```
// Display
cout << "KARTU MAKE IT 24"<<endl;
cout << inputString[0] << " " << inputString[1] << " " << inputString[2] << " " << inputString[3] << endl<<endl;
}

void getSolution(int input[], vector<string> *output){
    // KAMUS LOKAL
    int i,j,k,l,m,n,o, nSolution, nPermutation, ctr;
    ctr = 0;
    vector<Cards> allPermutation;
    // ALGORITMA
    functionPtr operators[4] = {tambah, kurang, kali, bagi};

    //Permutasi susunan kartu
    for (i= 0 ; i< 4; i++){
        for (j=0; j<4; j++){
            if (j != i){
                for (k=0; k<4; k++){
                    if (k!=j && k!=i){
                        for (l=0; l<4; l++){
                            if (l!=k && l!=j && l!=i){
                                Cards tempCard;
                                tempCard.card1 = input[i];
                                tempCard.card2 = input[j];
                                tempCard.card3 = input[k];
                                tempCard.card4 = input[l];
                                // Jika Belum Ada kartu pada susunan yang telah dieksekusi
                                if (!isExistCards(tempCard, allPermutation)){
                                    allPermutation.push_back(tempCard);
                                    // Permutasi operator
                                    for (m=0; m<=3; m++){
                                        for (n=0; n<=3; n++){
                                            for (o=0; o<=3; o++){
                                                functionPtr funcs[3] = {operators[m], operators[n], operators[o]};
                                                solveParentheses(tempCard,funcs,output); // cari solusi pada kemungkinan susunan tanda kurung
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Tugas Kecil 01 IF 2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
}
}
}
}
}
}
}
}
}
}
}

bool isValidInput(string inputLines)
{
    // KAMUS LOKAL
    string stringCards[4];
    string arg;
    int i, nValid, nArg;
    // ALGORITMA
    nValid = 0;
    nArg = 0;
    // Validasi Jumlah Input
    // splitting
    stringstream semiArgs(inputLines);
    while (getline(semiArgs, arg, ' ')) {
        if (nArg <= 3) {
            stringCards[i] = arg;
        }
        i++;
        nArg++;
    }

    if (nArg == 4) {
        // Validasi Kartu
        for (i = 0; i < 4; i++) {
            if (isValidCard(stringCards[i])) {
                nValid++;
            }
        }
    }

    return (nValid == 4); // Valid keempat kartunya
}

bool isValidCard(string stringCard)
{
    // KAMUS LOKAL
    string validCard[13] = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};
    int idx;
    bool valid;
    // ALGORITMA
    valid = false;
    idx = 0;
    // Cek apakah kartu terdapat di dalam list kartu yang valid
    while ((idx < 13) && (!valid))
    {
        if (stringCard == validCard[idx])
        {
            valid = true;
        }
        else {
            idx++;
        }
    }
    return valid;
}

bool isSameCards(Cards cards1, Cards cards2)
{
    // KAMUS LOKAL
    // ALGORITMA
    return (cards1.card1 == cards2.card1) && (cards1.card2 == cards2.card2) && (cards1.card3 == cards2.card3) && (cards1.card4 == cards2.card4);
}
```

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
bool isExistCards(Cards tempCard, vector<Cards> list)
{
    // KAMUS LOKAL
    bool exist;
    // ALGORITMA
    auto idx = list.begin();
    auto ends = list.end();
    exist = false;
    while ((idx != ends) && !exist){
        if (isSameCards(tempCard, *idx)){
            exist = true;
        } else {
            idx++;
        }
    }
    return exist;
}

void solveParentheses(Cards values, functionPtr funcs[], vector<string> *output)
{
    // KAMUS LOKAL
    double epsilon = pow(10.0, -12); // Epsilon untuk menangani galat akibat pecahan
    Operasi ops1, ops2, ops3;
    int val1, val2, val3, val4;
    functionPtr op1, op2, op3;
    double finalRes;
    // ALGORITMA
    // Setup
    val1 = values.card1;
    val2 = values.card2;
    val3 = values.card3;
    val4 = values.card4;

    op1 = funcs[0];
    op2 = funcs[1];
    op3 = funcs[2];
}
```

Tugas Kecil 01 IF 2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
// (val1 op1 val2) op2 (val3 op3 val4) => (a+b) + (c+d)
ops1 = op1(val1, val2);
ops2 = op3(val3, val4);
ops3 = op2(ops1.res, ops2.res);
if (abs(ops3.res-24)<epsilon){
    output->push_back("(" + to_string(val1) + " " + ops1.op + " " + to_string(val2) + ") " + " " + ops3.op + " " + "(" + to_string(val3) + " " + ops2.op + " " + to_string(val4) + ") " )
}

// ((val1 op1 val2) op2 val3) op3 val4 => ((a+b) + c) + d
ops2 = op2(ops1.res, val3);
ops3 = op3(ops2.res, val4);
if (abs(ops3.res-24)<epsilon){
    output->push_back("(" + to_string(val1) + " " + ops1.op + " " + to_string(val2) + ") " + " " + ops2.op + " " + to_string(val3) + ") " + " " + ops3.op + " " + to_string(val4));
}

// (val1 op1 (val2 op2 val3)) op3 val4 => (a+ (b+c)) + d
ops1 = op2(val2, val3);
ops2 = op1(val1, ops1.res);
ops3 = op3(ops2.res, val4);
if (abs(ops3.res-24)<epsilon){
    output->push_back("(" + to_string(val1) + " " + ops2.op + " " + "(" + to_string(val2) + " " + ops1.op + " " + to_string(val3) + ") " + " " + ops3.op + " " + to_string(val4));
}

// val1 op1 ((val2 op2 val3) op3 val4) => a + ((b+c)+d)
ops2 = op3(ops1.res, val4);
ops3 = op1(val1, ops2.res);
if (abs(ops3.res-24)<epsilon){
    output->push_back(to_string(val1) + " " + ops3.op + " " + "(" + to_string(val2) + " " + ops1.op + " " + to_string(val3) + ") " + " " + ops2.op + " " + to_string(val4)+")");
}

// val1 op1 (val2 op2 (val3 op3 val4)) => a + (b+(c+d))
ops1 = op3(val3, val4);
ops2 = op2(val2, ops1.res);
ops3 = op1(val1, ops2.res);
if (abs(ops3.res-24)<epsilon){
    output->push_back(to_string(val1) + " " + ops3.op + " " + "(" + to_string(val2) + " " + ops2.op + " " + "(" + to_string(val3) + " " + ops1.op + " " + to_string(val4)+") " + " " + ")");
}
}
```

```
void printSolution(vector<string> solutions)
{
    // KAMUS LOKAL
    // ALGORITMA
    for (auto solution = solutions.begin(); solution != solutions.end(); solution++)
    {
        cout << *solution << endl;
    }
}

void saveFile(string namaFile, vector<string> solutions)
{
    // KAMUS LOKAL
    // ALGORITMA
    // Create and Open a text file
    ofstream SolutionFile("test/" + namaFile + ".txt");
    string solutionTeks="";
    // Write to file
    for (auto solution = solutions.begin(); solution != solutions.end(); solution++)
    {
        solutionTeks += (*solution+ "\n");
    }
    SolutionFile << solutionTeks;
    // Close the file
    SolutionFile.close();
}
```

BAB IV : Eksperimen

A. Test Case 1 (2,5,A,K)

```
==== MAKE IT 24 ====

Tipe Input
1. Manual
2. Auto-Generated
Masukkan Jenis Input yang digunakan(1/2): 1

Masukkan 4 Kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dengan dipisahkan oleh satu spasi:
2 5 A K Q
Masukan Tidak Sesuai
Masukkan 4 Kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dengan dipisahkan oleh satu spasi:
2 5 A Z
Masukan Tidak Sesuai
Masukkan 4 Kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dengan dipisahkan oleh satu spasi:
2 5 A
Masukan Tidak Sesuai
Masukkan 4 Kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dengan dipisahkan oleh satu spasi:
2 5 A K
KARTU MAKE IT 24
2 5 A K
```

Banyak Solusi: 28	Solusi:
$(2 * 5) + (1 + 13)$	$1 + ((5 * 2) + 13)$
$((2 * 5) + 1) + 13$	$(1 + 13) + (2 * 5)$
$(2 * 5) + (13 + 1)$	$1 + (13 + (2 * 5))$
$((2 * 5) + 13) + 1$	$(1 + 13) + (5 * 2)$
$(2 + 1) * (13 - 5)$	$1 + (13 + (5 * 2))$
$(5 * 2) + (1 + 13)$	$(13 + (2 * 5)) + 1$
$((5 * 2) + 1) + 13$	$13 + ((2 * 5) + 1)$
$(5 * 2) + (13 + 1)$	$(13 + (5 * 2)) + 1$
$((5 * 2) + 13) + 1$	$13 + ((5 * 2) + 1)$
$(1 + (2 * 5)) + 13$	$(13 - 5) * (2 + 1)$
$1 + ((2 * 5) + 13)$	$(13 - 5) * (1 + 2)$
$(1 + 2) * (13 - 5)$	$(13 + 1) + (2 * 5)$
$(1 + (5 * 2)) + 13$	$13 + (1 + (2 * 5))$
	$(13 + 1) + (5 * 2)$
	$13 + (1 + (5 * 2))$

```
Apakah ingin menyimpan solusi? (y/n):
y
Masukkan Nama File Solusi:
testCase1

Execution Time: 0 s
```


Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```
Tucil1_13521066 > test > testCase1.txt
You, 19 seconds ago | 1 author (You)
1 (2 * 5) + (1 + 13)
2 ((2 * 5) + 1) + 13
3 (2 * 5) + (13 + 1)
4 ((2 * 5) + 13) + 1
5 (2 + 1) * (13 - 5)
6 (5 * 2) + (1 + 13)
7 ((5 * 2) + 1) + 13
8 (5 * 2) + (13 + 1)
9 ((5 * 2) + 13) + 1
10 (1 + (2 * 5)) + 13
11 1 + ((2 * 5) + 13)
12 (1 + 2) * (13 - 5)
13 (1 + (5 * 2)) + 13
14 1 + ((5 * 2) + 13)
15 (1 + 13) + (2 * 5)
16 1 + (13 + (2 * 5))
17 (1 + 13) + (5 * 2)
18 1 + (13 + (5 * 2))
19 (13 + (2 * 5)) + 1
20 13 + ((2 * 5) + 1)
21 (13 + (5 * 2)) + 1
22 13 + ((5 * 2) + 1)
23 (13 - 5) * (2 + 1)
24 (13 - 5) * (1 + 2)
25 (13 + 1) + (2 * 5)
26 13 + (1 + (2 * 5))
27 (13 + 1) + (5 * 2)
28 13 + (1 + (5 * 2))
```

B. TestCase2 (2 2 4 J)

```
Tipe Input
1. Manual
2. Auto-Generated
Masukkan Jenis Input yang digunakan(1/2): 1

Masukkan 4 Kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dengan dipisahkan oleh satu spasi:
2 2 4 J
KARTU MAKE IT 24
2 2 4 J
```

Banyak Solusi: 42	$2 + (11 * (4 / 2))$	$(4 + (11 * 2)) - 2$
Solusi:	$(2 * 11) + (4 - 2)$	$4 + ((11 * 2) - 2)$
$2 - ((2 - 4) * 11)$	$((2 * 11) + 4) - 2$	$((4 * 11) / 2) + 2$
$2 + ((4 - 2) * 11)$	$(2 * 11) + (4 / 2)$	$(4 * (11 / 2)) + 2$
$2 + ((4 / 2) * 11)$	$(4 - 2) + (2 * 11)$	$((11 * 2) - 2) + 4$
$2 + (4 / (2 / 11))$	$4 - (2 - (2 * 11))$	$(11 * 2) - (2 - 4)$
$2 + ((4 * 11) / 2)$	$(4 / 2) + (2 * 11)$	$(11 * 2) + (4 - 2)$
$2 + (4 * (11 / 2))$	$(4 + (2 * 11)) - 2$	$((11 * 2) + 4) - 2$
$2 + ((11 / 2) * 4)$	$4 + ((2 * 11) - 2)$	$(11 * 2) + (4 / 2)$
$2 + (11 / (2 / 4))$	$(4 - 2) + (11 * 2)$	$((11 / 2) * 4) + 2$
$2 - (11 * (2 - 4))$	$4 - (2 - (11 * 2))$	$(11 / (2 / 4)) + 2$
$((2 * 11) - 2) + 4$	$((4 - 2) * 11) + 2$	$(11 * (4 - 2)) + 2$
$(2 * 11) - (2 - 4)$	$(4 / 2) + (11 * 2)$	$((11 * 4) / 2) + 2$
$2 + (11 * (4 - 2))$	$((4 / 2) * 11) + 2$	$(11 * (4 / 2)) + 2$
$2 + ((11 * 4) / 2)$	$(4 / (2 / 11)) + 2$	

```
Apakah ingin menyimpan solusi? (y/n):
n
```

```
Execution Time: 0 s
```

Tugas Kecil 01 IF 2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

C. Test Case 3 (6, 6, 6, 6)

```
==== MAKE IT 24 ====


Tipe Input
1. Manual
2. Auto-Generated
Masukkan Jenis Input yang digunakan(1/2): 1

Masukkan 4 Kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dengan dipisahkan oleh satu spasi:
6 6 6 6
KARTU MAKE IT 24
6 6 6 6

Banyak Solusi: 7
Solusi:
(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

Apakah ingin menyimpan solusi? (y/n):
y
Masukkan Nama File Solusi:
testCase3

Execution Time: 0 s
```

```
Tucil1_13521066 > test >  testCase3.txt
1  (6 + 6) + (6 + 6)
2  ((6 + 6) + 6) + 6
3  (6 + (6 + 6)) + 6
4  6 + ((6 + 6) + 6)
5  6 + (6 + (6 + 6))
6  (6 * 6) - (6 + 6)
7  ((6 * 6) - 6) - 6
```

D. Test Case 4 (A, A, A, A)

```
==== MAKE IT 24 ====

Tipe Input
1. Manual
2. Auto-Generated
Masukkan Jenis Input yang digunakan(1/2): 1

Masukkan 4 Kartu {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K} dengan dipisahkan oleh satu spasi:
A A A A
KARTU MAKE IT 24
A A A A

Tidak Ada Solusi

Execution Time: 0.001 s
```

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

E. Test Case 5 Random(4, 10, 3, Q)

```
===== MAKE IT 24 =====
```

Tipe Input

1. Manual

2. Auto-Generated

Masukkan Jenis Input yang digunakan(1/2): 2

KARTU MAKE IT 24

4 10 3 Q

Banyak Solusi: 27

Solusi:

```
4 * (10 - (12 / 3))
((4 * 3) - 10) * 12
(4 * 3) * (12 - 10)
4 * (3 * (12 - 10))
(4 * (12 - 10)) * 3
4 * ((12 - 10) * 3)
((10 - 4) / 3) * 12
(10 - 4) / (3 / 12)
(10 - 4) * (12 / 3)
((10 - 4) * 12) / 3
```

```
(10 - (12 / 3)) * 4
((3 * 4) - 10) * 12
(3 * 4) * (12 - 10)
3 * (4 * (12 - 10))
(3 * (12 - 10)) * 4
3 * ((12 - 10) * 4)
12 * ((4 * 3) - 10)
(12 - 10) * (4 * 3)
((12 - 10) * 4) * 3
(12 * (10 - 4)) / 3
12 * ((10 - 4) / 3)
(12 - 10) * (3 * 4)
((12 - 10) * 3) * 4
12 * ((3 * 4) - 10)
12 / (3 - (10 / 4))
(12 / 3) * (10 - 4)
12 / (3 / (10 - 4))
```

Tucil1_13521066 > test > testCase5.txt

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

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

F. Test Case 6 Random(10, 7, J, J)

```
==== MAKE IT 24 ====

Tipe Input
1. Manual
2. Auto-Generated
Masukkan Jenis Input yang digunakan(1/2): 2

KARTU MAKE IT 24
10 7 J J

Tidak Ada Solusi

Execution Time: 0 s
```

G. Test Case 7 Random(10, 7, J, J)

```
==== MAKE IT 24 ====

Tipe Input
1. Manual
2. Auto-Generated
Masukkan Jenis Input yang digunakan(1/2): 2

KARTU MAKE IT 24
10 A 10 4
```

Tugas Kecil 01 IF 2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

```

Banyak Solusi: 94
Solusi:
(10 + (1 * 10)) + 4
10 + ((1 * 10) + 4)
10 + (1 * (10 + 4))
(10 * 1) + (10 + 4)
((10 * 1) + 10) + 4
(10 / 1) + (10 + 4)
((10 / 1) + 10) + 4
(10 + (1 * 4)) + 10
10 + ((1 * 4) + 10)
10 + (1 * (4 + 10))
(10 * 1) + (4 + 10)
((10 * 1) + 4) + 10
(10 / 1) + (4 + 10)
((10 / 1) + 4) + 10
(10 + 10) + (1 * 4)
10 + (10 + (1 * 4))
((10 + 10) * 1) + 4
(10 + (10 * 1)) + 4
10 + ((10 * 1) + 4)
((10 + 10) / 1) + 4
(10 + (10 / 1)) + 4
10 + ((10 / 1) + 4)
((10 + 10) * 1)
(10 + 10) + (4 * 1)
((10 + 10) + 4) * 1
(10 + (10 + 4)) * 1
10 + ((10 + 4) * 1)
10 + (10 + (4 * 1))
(10 + 10) + (4 / 1)
((10 + 10) + 4) / 1
(10 + (10 + 4)) / 1
10 + ((10 + 4) / 1)
10 + (10 + (4 / 1))
((10 * 10) / 4) - 1
(10 * (10 / 4)) - 1
(10 + 4) + (1 * 10)
10 + (4 + (1 * 10))
((10 + 4) * 1) + 10
(10 + (4 * 1)) + 10
10 + ((4 * 1) + 10)
((10 + 4) / 1) + 10
((10 + (4 / 1)) + 10)
10 + ((4 / 1) + 10)
(10 + 4) + (10 * 1)
((10 + 4) + 10) * 1
(10 + 4) + (10 / 1)
((10 + 4) + 10) / 1
(10 + (4 + 10)) / 1
10 + ((4 + 10) / 1)
10 + (4 + (10 / 1))
((10 / 4) * 10) - 1
(10 / (4 / 10)) - 1
(1 * 10) + (10 + 4)
((1 * 10) + 10) + 4
(1 * (10 + 10)) + 4
1 * ((10 + 10) + 4)
1 * (10 + (10 + 4))
(1 * 10) + (4 + 10)
((1 * 10) + 4) + 10
(1 * (10 + 4)) + 10
1 * ((10 + 4) + 10)
(1 * 10) + (10 + 10)
(1 * 10) + (10 + 10)
(4 + (10 + 10)) * 1
(4 + (10 + 10)) * 1
4 + ((10 + 10) * 1)
4 + (10 + (10 * 1))
(4 + 10) + (10 / 1)
(4 + (1 * 10)) + 10
4 + ((1 * 10) + 10)
4 + (1 * (10 + 10))
(4 * 1) + (10 + 10)
((4 * 1) + 10) + 10
(4 / 1) + (10 + 10)
((4 / 1) + 10) + 10

```

H. Test Case 7 Random(Q, J, 6, A)

```

==== MAKE IT 24 ====
1. Manual
2. Auto-Generated
Masukkan Jenis Input yang digunakan(1/2): 2

KARTU MAKE IT 24
Q J 6 A

Banyak Solusi: 16
Solusi:
(12 * (11 + 1)) / 6
12 * ((11 + 1) / 6)
(12 / 6) * (11 + 1)
12 / (6 / (11 + 1))
(12 / 6) * (1 + 11)
12 / (6 / (1 + 11))
(12 * (1 + 11)) / 6
12 * ((1 + 11) / 6)
(11 + 1) * (12 / 6)
((11 + 1) * 12) / 6
((11 + 1) / 6) * 12
(11 + 1) / (6 / 12)
(1 + 11) * (12 / 6)
((1 + 11) * 12) / 6
((1 + 11) / 6) * 12
(1 + 11) / (6 / 12)

Apakah ingin menyimpan solusi? (y/n):
n

Execution Time: 0 s

```

BAB V : *LINK REPOSITORY*

https://github.com/Mehmed13/Tucil1_13521066

BAB VI : CHECKING TABLE

POIN	YA	TIDAK
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil <i>running</i>	✓	
3. Program dapat membaca input / <i>generate</i> sendiri dan memberikan luaran	✓	
4. Solusi yang diberikan program memenuhi (berhasil mencapai 24)	✓	
5. Program dapat menyimpan solusi dalam file teks	✓	