Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*

Disusun oleh:
Juan Christopher Santoso
13521116

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

Daftar Isi

Deskripsi Permasalahan **1**

Penjelasan Algoritma Brute Force **2**
A. Definisi Algoritma Brute Force 2
B. Algoritma Brute Force pada Program Penyelesaian Permainan Kartu 24 2
1. Kombinasi pada Operator Aritmatika 2
2. Kombinasi pada Pengurungan 3
3. Kombinasi pada Urutan Posisi Angka 3

Pemaparan dan Penjelasan Source Code **5**
A. Pemaparan Source Code pada File main.cpp 5
B. Pemaparan Source Code pada File io.cpp 11
C. Pemaparan Source Code pada File calc.cpp 13
D. Penjelasan Singkat Source Code 16

Implementasi Program **17**
A. Screenshot Pelaksanaan Program 17
B. Rekapitulasi Percobaan 23

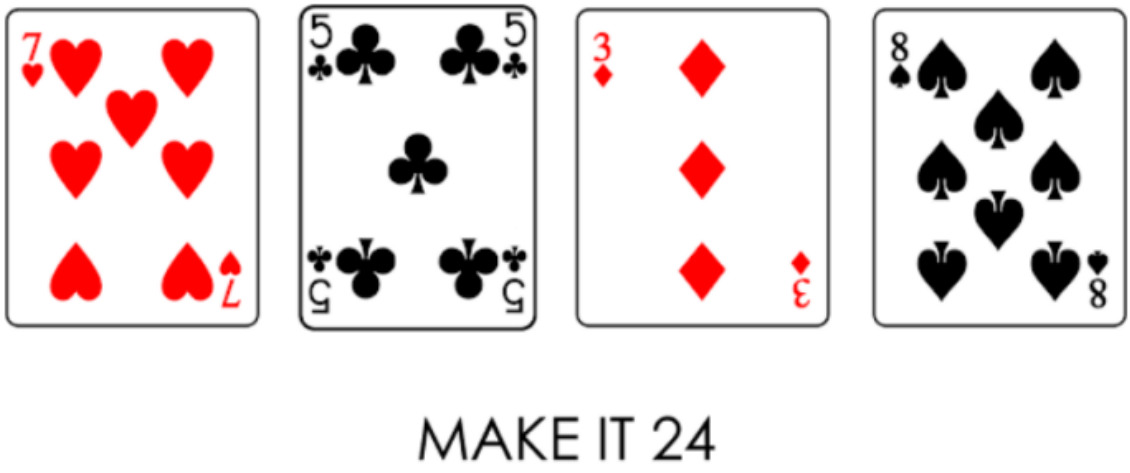
Penutup **25**

Daftar Pustaka **26**

Lampiran **27**
A. Tabel Progress Checklist 27
B. Github Repository Link 27

Bab I
Deskripsi Permasalahan

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.



Gambar 1.1 Permainan 24 Menggunakan Kartu Remi

Bab II
Penjelasan Algoritma *Brute Force*

A. Definisi Algoritma *Brute Force*

Frasa *brute force* berasal dari Bahasa Inggris yang dapat dipecah menjadi kata *brute* dan *force.* Kata *brute* memiliki arti ketiadaan kecerdasan (*absence of reasoning or intelligence*). Di sisi lain, kata *force* memiliki arti yang sama dengan paksaan (*coercion or compulsion*). Maka dari itu, *brute force* dapat diartikan sebagai suatu paksaan tanpa adanya kecerdasan pemikiran.

Algoritma *brute force* umumnya berkaitan dengan cara penyelesaian permasalahan yang tidak memikirkan efisiensi dan efektivitas. Mencoba segala kemungkinan yang ada sehingga ditemukan solusi yang diinginkan dapat menjelaskan bagaimana algoritma *brute force* bekerja. Algoritma *brute force* adalah pendekatan yang paling *straightforward* dalam memecahkan suatu persoalan. "*Just do it!"* dan "*Just solve it!"* merupakan kalimat yang dapat menggambarkan algoritma tersebut.

B. Algoritma *Brute Force* pada Program Penyelesaian Permainan Kartu 24

Dalam penyelesaian permainan kartu 24, algoritma *brute force* yang digunakan adalah dengan mencoba segala kombinasi yang bisa dibentuk menggunakan 4 kartu yang digunakan. Terdapat tiga aspek yang perlu diperhatikan dalam menentukan segala kemungkinan yang dapat dibentuk, yaitu operator aritmatika yang digunakan, peletakkan tanda kurung (pengurungan), dan urutan posisi angka dalam perhitungan.

1. Kombinasi pada Operator Aritmatika

   Mengingat bahwa pemain memiliki 4 kartu pada permainan kartu 24 dan setiap kartu dapat digunakan tepat sekali, maka terdapat 3 *slots* dimana operator aritmatika dapat ditempatkan untuk melakukan perhitungan. Untuk setiap operator yang digunakan, terdapat 4 pilihan operator yang *valid* untuk dipakai, yaitu operator penambahan (+), operator pengurangan (-), operator perkalian (*), dan operator pembagian (/). Dalam penggunaan operator, tidak ada larangan bahwa operator tidak boleh dipakai lebih dari sekali. Dengan begitu, banyaknya kombinasi yang dapat dibentuk dari penggunaan operator aritmatika adalah 64 kombinasi. Hal ini dapat ditentukan dengan menghitung 4x4x4 kemungkinan.

   Daftar dari kemungkinan kombinasi operator aritmatika dapat dilihat pada tabel berikut:

Tabel 2.1 Tabel Daftar Kemungkinan Kombinasi Operator Aritmatika

+ + +	+ + -	+ + *	+ + /	+ - +	+ - -	+ - *	+ - /
+ * +	+ * -	+ * *	+ * /	+ / +	+ / -	+ / *	+ / /

- + +	- + -	- + *	- + /	- - +	- - -	- - *	- - /
- * +	- * -	- * *	- * /	- / +	- / -	- / *	- / /
* + +	* + -	* + *	* + /	* - +	* - -	* - *	* - /
* * +	* * -	* * *	* * /	* / +	* / -	* / *	* / /
/ + +	/ + -	/ + *	/ + /	/ - +	/ - -	/ - *	/ - /
/ * +	/ * -	/ * *	/ * /	/ / +	/ / -	/ / *	/ / /

2. Kombinasi pada Pengurungan

   Terdapat lima kemungkinan peletakkan tanda kurung dalam melakukan perhitungan 4 angka. Misalkan nilai setiap kartu secara berturut-turut adalah A, B, C, dan D dan tanda operator dimisalkan dengan huruf ‘x’, maka daftar dari kombinasi peletakkan tanda kurung dapat dituliskan sebagai berikut:

   a. ((A x B) x C) x D
   b. (A x (B x C)) x D
   c. (A x B) x (C x D)
   d. A x ((B x C) x D)
   e. A x (B x (C x D))

3. Kombinasi pada Urutan Posisi Angka

   Terdapat 4 angka yang dapat digunakan mencari nilai 24 pada permainan kartu 24. Namun, tidak ada aturan yang mengatur mengenai penempatan angka dalam melakukan perhitungan. Maka dari itu, menentukan pasangan angka mana yang dihitung terlebih dahulu bergantung kepada kecerdasan dan kreativitas pemain. Meski tidak ada aturan yang mengatur penempatan angka dalam melakukan kalkulasi, perlu diperhatikan bahwa terdapat aturan dimana setiap angka yang dimiliki harus digunakan dan digunakan tepat sekali. Dengan begitu, banyaknya kombinasi yang dapat dibentuk dari aspek urutan posisi angka adalah 24 kombinasi. Hal ini dapat ditentukan dengan menghitung 4x3x2x1 kemungkinan.

   Asumsikan setiap angka yang dimiliki dimisalkan dengan angka 1, 2, 3, dan 4, maka daftar dari kemungkinan urutan posisi angka dapat dilihat pada tabel berikut:

Tabel 2.2 Tabel Daftar Kemungkinan Kombinasi Urutan Posisi Angka

1 2 3 4	1 2 4 3	1 3 2 4	1 3 4 2	1 4 2 3	1 4 3 2
2 1 3 4	2 1 4 3	2 3 1 4	2 3 4 1	2 4 1 3	2 4 3 1

3 1 2 4	3 1 4 2	3 2 1 4	3 2 4 1	3 4 1 2	3 4 2 1
4 1 2 3	4 1 3 2	4 2 1 3	4 2 3 1	4 3 1 2	4 3 2 1

Setelah membahas semua jumlah kombinasi yang dapat dibentuk dari ketiga aspek tersebut, maka dapat ditentukan bahwa jumlah maksimal kemungkinan yang dibentuk program dalam menyelesaikan permasalahan permainan kartu 24 adalah 7680 kemungkinan. Hal ini ditentukan dari perkalian jumlah kombinasi dari setiap aspek yang telah dibahas yaitu, 64 kombinasi penggunaan operator aritmatika, 5 kombinasi peletakan tanda kurung, dan 24 kombinasi penempatan urutan posisi angka. Dalam hal ini, kembali ditekankan bahwa 7680 kemungkinan adalah jumlah kemungkinan maksimal yang dibentuk oleh program. Tidak dalam semua kasus program membentuk 7680 kemungkinan. Hal ini dikarenakan terdapat kondisi dimana angka-angka yang dimiliki bisa saja bernilai sama, contohnya kombinasi kartu 5-5-A-J yang memiliki 2 buah angka 5 atau bahkan kombinasi kartu 8-8-8-8 yang semua kartunya bernilai 8. Asumsikan kasus dimana kombinasi kartu yang dimiliki adalah 5-5-A-J dan kedua angka 5 tersebut dapat dibedakan sebagai 5(♠) dan 5(♥). Dalam hal ini, urutan posisi 5(♠)-5(♥)-A-J bernilai sama dengan urutan posisi 5(♥)-5(♠)-A-J. Maka dari itu, jumlah kombinasi yang dibentuk berjumlah kurang dari 24.

Bab III
Pemaparan dan Penjelasan Source Code

Program yang dirancang untuk menyelesaikan permasalahan permainan kartu 24 ini dibuat dalam bahasa pemrograman C++.

A. Pemaparan *Source Code* pada File *main.cpp*

```
#include <iostream>
#include <chrono>
#include "calc.cpp"
#include "io.cpp"
using namespace std;
using namespace std::chrono;

int main () {
    // Variables Declaration
    int i, j , k;
    int w, x, y, z;
    int arr[4];
    int allValid = 1;
    int nSolutions = 0;
    string tempOpr;
    string arrSolutions[500];
    int permuteArr[24][4];
    int nPermutes = 0;

    // Display Ascii Art
    displayArt();

    // Reading Input
    char ans;
    float temp;
    cout << "Hi there! Welcome to 24 card game solver! " << endl;
    cout << "Here are the options you can use to run the program: " << endl;
    cout << "(1) to manually insert your desired numbers. " << endl;
    cout << "(2) to automatically generate random numbers. " << endl;
    cout << "=> " ;
    cin >> ans;
    cin.clear();
    cin.ignore(100,'\n');

    while (ans > '2' || ans < '1'){
        cout << "The answer you just inputted is invalid. Please insert an appropriate
input." << endl;
        cout << "=> " ;
```

```
        cin >> ans;
        cin.clear();
        cin.ignore(100,'\n');
    }

    if (ans == '1'){
        cout << "Please insert the desired numbers. List of valid inputs:" << endl;
        cout << "A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K. (e.g. 10 A 3 J)" << endl;
        string a, b, c, d;
        cout << "=> " ;
        cin >> a >> b >> c >> d;
        cin.clear();
        cin.ignore(100,'\n');

        arr[0] = integerConvert(a);
        arr[1] = integerConvert(b);
        arr[2] = integerConvert(c);
        arr[3] = integerConvert(d);

        for (i = 0; i < 4; i++){
            if (arr[i] == -999){
                allValid = 0;
            }
        }

        // Error handling
        while (!allValid){
            cout << "The program has detected (an) invalid input(s). Please insert the
suitable ones." << endl;
            allValid = 1;
            cout << "=> " ;
            cin >> a >> b >> c >> d;
            cin.clear();
            cin.ignore(100,'\n');

            arr[0] = integerConvert(a);
            arr[1] = integerConvert(b);
            arr[2] = integerConvert(c);
            arr[3] = integerConvert(d);

            for (i = 0; i < 4; i++){
                if (arr[i] == -999){
                    allValid = 0;
                }
            }
        }
```

```
    } else {
        // Random Number Generator
        cout << "The program will generate random inputs for you." << endl;

        srand(time(0));
        for (i = 0 ; i < 4; i++){
            arr[i] = (rand() % 13) +1;
        }
    }

    cout << "Here are your numbers : ";
    printArr(arr);

    // Solving Algorithm
    cout << "The program starts to solve..." << endl;

    // Starting the timer
    auto start = high_resolution_clock::now();

    for (w = 0; w < 4; w++){
        for (x = 0; x < 4 ; x++){
            for (y = 0; y < 4; y++){
                for (z = 0; z< 4; z++){
                    if (w != x && w != y && w != z && x != y && x != z && y != z){
                        int newArr[4];
                        newArr[0] = arr[w];
                        newArr[1] = arr[x];
                        newArr[2] = arr[y];
                        newArr[3] = arr[z];

                        // Make sure multiple same numbers are not processed multiple
times
                        if (!isUsedPermute(newArr, permuteArr, nPermutes)){

                            // Every valid permutation will be stored in the array
                            // the index of permutation array will increase
                            for (i = 0; i < 4; i++){
                                permuteArr[nPermutes][i] = newArr[i];
                            }
                            nPermutes++;

                            // 4x4x4 loops represents 4 options for 3 operations that
are used in calculations
                            for (i = 0; i < 4 ; i++){
                                for (j = 0; j < 4 ; j++){
                                    for (k = 0; k < 4; k++){
                                        temp = calculate1(i,j,k, newArr);
```

```
                                    // cout << temp << "|1|" <<endl; // Hanya
untuk mengecek
                                    if (temp == 24){
                                        tempOpr = makeOpr(i, j, k, newArr, 1);
                                        // cout << tempOpr << endl;
                                        arrSolutions[nSolutions] = tempOpr;
                                        if(nSolutions < 500){
                                            nSolutions++;
                                        }
                                    }
                                    temp = calculate2(i,j,k, newArr);
                                    //cout << temp << "|2|" <<endl; // Hanya untuk
mengecek
                                    if (temp == 24){
                                        tempOpr = makeOpr(i, j, k, newArr, 2);
                                        // cout << tempOpr << endl;
                                        arrSolutions[nSolutions] = tempOpr;
                                        if(nSolutions < 500){
                                            nSolutions++;
                                        }
                                    }
                                    temp = calculate3(i,j,k, newArr);
                                    // cout << temp << "|3|" <<endl; // Hanya
untuk mengecek
                                    if (temp == 24){
                                        tempOpr = makeOpr(i, j, k, newArr, 3);
                                        // cout << tempOpr << endl;
                                        arrSolutions[nSolutions] = tempOpr;
                                        if(nSolutions < 500){
                                            nSolutions++;
                                        }
                                    }
                                    temp = calculate4(i,j,k, newArr);
                                    //cout << temp << "|4|" <<endl; // Hanya untuk
mengecek
                                    if (temp == 24){
                                        tempOpr = makeOpr(i, j, k, newArr, 4);
                                        // cout << tempOpr << endl;
                                        arrSolutions[nSolutions] = tempOpr;
                                        if(nSolutions < 500){
                                            nSolutions++;
                                        }
                                    }
                                    temp = calculate5(i,j,k, newArr);
                                    // cout << temp << "|5|" <<endl; // Hanya
untuk mengecek
```

```
                                        if (temp == 24){
                                            tempOpr = makeOpr(i, j, k, newArr, 5);
                                            // cout << tempOpr << endl;
                                            arrSolutions[nSolutions] = tempOpr;
                                            if(nSolutions < 500){
                                                nSolutions++;
                                            }
                                        }
                                    }
                                }
                            }


                        }
                    }
                }
            }
        }

    // Stopping the timer1 (calculation only)
    auto stop1 = high_resolution_clock::now();

    // Displaying Solutions
    string hBorder;
    for (i = 0; i < (tempOpr.length()+8) ; i++){
        hBorder += "=";
    }
    cout << "The amount of solutions: " << nSolutions << endl;


    if (nSolutions == 0){
        cout << " []============================================================[]" <<
endl;
        cout << " || No equation can be showed since there is no solution. ||" <<
endl;
        cout << " []============================================================[]" <<
endl;
    } else {
        cout << "  []"<< hBorder <<"[]  " << endl;
        cout << "           Solutions" << endl;
        cout << "  []"<< hBorder <<"[]  " << endl;
        for (i = 0; i < nSolutions; i++){
            cout << "  ||     " <<arrSolutions[i] << "    ||  " << endl;
        }
        cout << "  []"<< hBorder <<"[]  " << endl;
    }
```

```
    // Stopping the timer2 (calculation + displaying result)
    auto stop2 = high_resolution_clock::now();

    auto duration1 = duration_cast<microseconds>(stop1-start);
    auto duration2 = duration_cast<microseconds>(stop2-start);

    cout << "Execution time for BF Algorithm only: " << duration1.count() << "
microseconds" << endl;
    cout << "Execution time including displaying the result: " << duration2.count() <<
" microseconds" << endl;

    // Saving the results in a file
    char save;
    cout << "Do you wish to save the result in a file? (y/n)" << endl ;
    do {
        cout << "=> " ;
        cin >> save;
        cin.clear();
        cin.ignore(100,'\n');
        save = capitalize(save);
        if (save != 'Y' && save != 'N'){
            cout << "Invalid input detected. Please insert again: " << endl;
        }
    } while (save != 'Y' && save != 'N');

    if (save == 'Y'){
        string fileName;
        cout << "Please insert the file name: " << endl;
        cout << "=> " ;
        cin >> fileName;
        writeOnFile(arrSolutions, fileName, nSolutions);
        cout << fileName << ".txt has been saved and stored inside the 'test' folder!"
<< endl;
    }

    cout << "The program has been stopped." << endl;

    return 0;
}
```

B. Pemaparan *Source Code* pada File io.*cpp*

```cpp
#include <iostream>
#include <fstream>
using namespace std;

void displayArt(){
    string bufferline;
    ifstream file("../src/art.txt");
    while(getline(file, bufferline)){
        cout << bufferline << endl;
    }
    file.close();
}

char capitalize(char C){
    // Digunakan untuk memastikan bahwa input user untuk Char adalah huruf Kapital
    if ( C >= 97 && C <= 122){
        return (C-32);
    }
}

int integerConvert(string S){
    // Digunakan untuk mengubah nilai A menjadi 1, J menjadi 11, Q menjadi 12, dan K
menjadi 13
    if (S.length() > 2){
        return -999;
    } else if (S.length() == 2){
        if (S[0] == '1' && S[1] == '0'){
            return 10;
        } else {
            return -999;
        }
    } else {
        char temp = capitalize(S[0]);
        if ( temp >= 50 && temp <= 57){ // Angka
            return (temp-48);
        } else {
            if (temp == 65){ // Ace
                return 1;
            } else if ( temp == 74){ // Jack
                return 11;
            } else if (temp == 81){ // Queen
                return 12;
            } else if (temp == 75){ // King
                return 13;
            } else { // Untuk kasus dimana input tidak valid
                return -999;
```

```
            }
        }
    }

}

void printArr(int arr[4]){
    // Melakukan print atas nilai-nilai yang dikandung pada array
    int i;
    for( i = 0 ; i < 4; i++){
        switch (arr[i]){
            case 1:
                cout << 'A' << ' ';
                break;
            case 11:
                cout << 'J'<< ' ';
                break;
            case 12:
                cout << 'Q'<< ' ';
                break;
            case 13:
                cout << 'K'<< ' ';
                break;
            default:
                cout << arr[i]<< ' ';
                break;
        }
    }
    cout << endl;
}

void writeOnFile(string strArr[500], string fileName, int n){
    ofstream outputFile;
    string path = "../test/"+fileName+".txt";
    outputFile.open(path);

    outputFile << n << endl;
    if (n == 0){
        outputFile << "No equation can be showed since there is no solution." << endl;
    } else {
        for (int i = 0; i < n; i++){
            outputFile << strArr[i] << endl;
        }
    }
    outputFile.close();
}
```

C. Pemaparan *Source Code* pada File *calc.cpp*

```cpp
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <time.h>
using namespace std;


float operate(int key, float a, float b){
    // Melakukan operasi dengan key yang telah ditentukan
    if (key == 0){
        return a + b;
    } else if (key == 1){
        return a - b ;
    } else if (key == 2){
        return a * b;
    } else {
        return a / b;
    }
}

int isSamePermute(int arr1[4], int arr2[4]){
    // Melakukan cek apakah sebuah array yang berisikan sebuah permutasi adalah array
yang sama
    // Digunakan untuk menunjang pengecekan permutasi berulang
    int i;
    for (i = 0; i < 4; i++){
        if (arr1[i] != arr2[i]){
            return 0;
        }
    }
    return 1;
}

int isUsedPermute(int arr[4], int permArr[24][4], int n){
    // Melakukan cek apakah permutasi yang serupa telah dilakukan sebelumnya.
    // Hal ini digunakan untuk menghindari pengulangan untuk urutan angka yang serupa
    int i;
    int used;
    for (i = 0; i < n; i++){
        if (isSamePermute(arr, permArr[i])){
            return 1;
        }
    }
    return 0;
}
```

```
/*
5 jenis pengurungan
((a x b) y c) z d
( a x (b y c))z d
( a x b) y (c z d)
a x ((b y c) z d )
a x (b y (c z d ))

ket : a, b, c, d => angka (integer) | x, y, z => operator
*/

float calculate1(int x, int y, int z, int arr[4]){
    // Melakukan perhitungan jenis pengurungan 1
    return operate(z, (operate(y, (operate(x, arr[0], arr[1])), arr[2])), arr[3]);
}

float calculate2(int x, int y, int z, int arr[4]){
    // Melakukan perhitungan jenis pengurungan 2
    return operate(z, (operate(x, arr[0], (operate(y, arr[1], arr[2])))), arr[3]);
}

float calculate3(int x, int y, int z, int arr[4]){
    // Melakukan perhitungan jenis pengurungan 3
    return operate(y, (operate(x, arr[0], arr[1])) ,(operate(z, arr[2], arr[3])) );
}

float calculate4(int x, int y, int z, int arr[4]){
    // Melakukan perhitungan jenis pengurungan 4
    return operate(x, arr[0], (operate(z, (operate(y, arr[1], arr[2])), arr[3])));
}

float calculate5(int x, int y, int z, int arr[4]){
    // Melakukan perhitungan jenis pengurungan 5
    return operate(x, arr[0], (operate(y, arr[1], (operate(z, arr[2], arr[3])))));
}

char keyToOpr(int x){
    // Mengubah integer key menjadi sebuah char operasi
    switch (x){
        case 0:
            return '+';
            break;
        case 1:
            return '-';
            break;
        case 2:
            return '*';
```

```
            break;
        default:
            return '/';
            break;
    }
}

string makeOpr(int x, int y, int z, int arr[4], int type){
    // Membuat string operasi dengan pengurungan
    string res = "";
    char a = keyToOpr(x);
    char b = keyToOpr(y);
    char c = keyToOpr(z);
    if (type == 1){
        // ((a x b) y c) z d
        res += "((" + to_string(arr[0]) + a + to_string(arr[1]) + ')';
        res += b + to_string(arr[2]) + ')';
        res += c + to_string(arr[3]);
        return res;
    } else if (type == 2){
        // ( a x (b y c))z d
        res += '(' + to_string(arr[0]) + a + '(';
        res +=  to_string(arr[1]) + b + to_string(arr[2]) + "))";
        res += c + to_string(arr[3]);
        return res;
    } else if (type == 3){
        // ( a x b) y (c z d)
        res += '(' + to_string(arr[0]) + a + to_string(arr[1]) + ')';
        res += b;
        res +=  '('+ to_string(arr[2]) + c + to_string(arr[3]) + ')';
        return res;
    } else if (type == 4){
        // a x ((b y c) z d )
        res += to_string(arr[0])+a+"((";
        res += to_string(arr[1])+ b+ to_string(arr[2]) + ')';
        res += c + to_string(arr[3]) + ')';
        return res;
    } else {
        // a x (b y (c z d ))
        res += to_string(arr[0]) + a + '(';
        res += to_string(arr[1]) + b + '(';
        res += to_string(arr[2]) + c + to_string(arr[3]) + "))";
        return res;
    }
}
```

D. Penjelasan Singkat *Source Code*

Source Code terbagi menjadi 3 *file*. Pertama, *io.cpp* berisikan segala fungsi dan prosedur untuk mengatur input dan output yang diproses oleh program. Lalu, *calc.cpp* berisikan segala fungsi dan prosedur dalam melakukan perhitungan menggunakan algoritma *brute force*. Terakhir, *main.cpp* berisikan algoritma utama dalam menyelesaikan permasalahan permainan kartu 24.

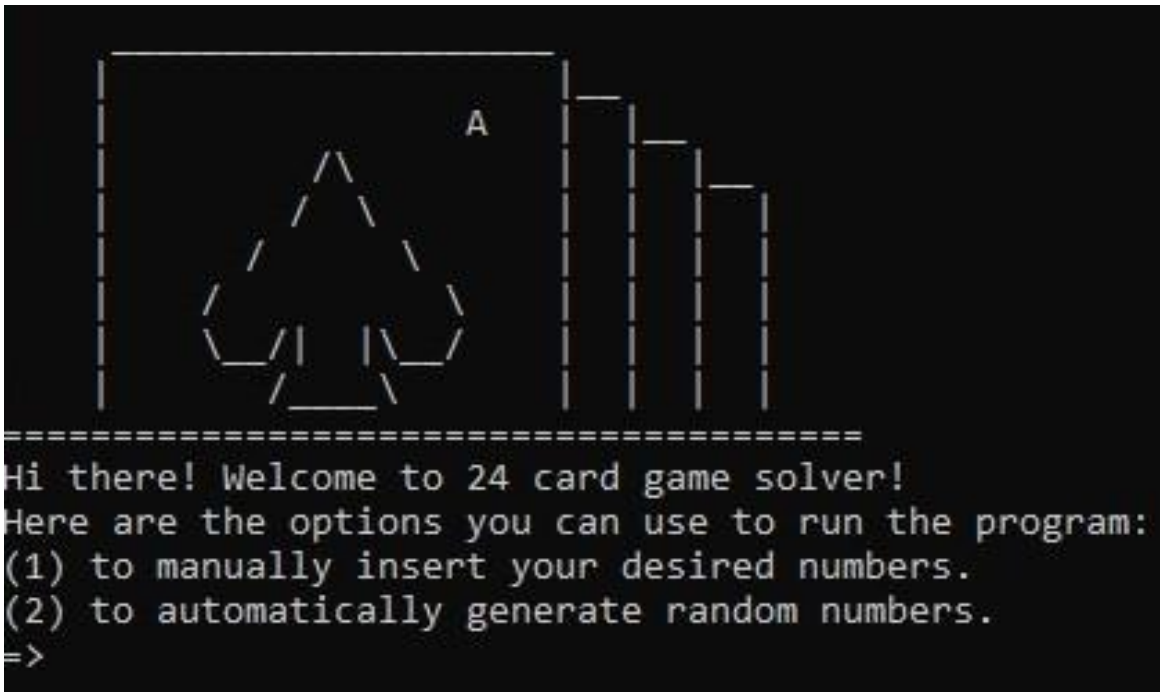
Langkah-langkah penyelesaian yang dilakukan oleh program dapat dijelaskan secara singkat sebagai berikut:

1. Menampilkan *welcome screen* kepada *user* dan meminta *input* dari *user* mengenai apakah sang *user* ingin secara manual memasukkan kombinasi kartu yang diinginkan atau meminta program secara otomatis menghasilkan kombinasi kartunya sendiri.
2. Bila *user* ingin memasukkan kombinasi kartu secara manual, program akan terus meminta *input* dari user sampai *input* dari user tersebut merupakan kombinasi kartu yang valid.
3. Kombinasi kartu yang telah dimasukkan atau dihasilkan akan disimpan di dalam sebuah *array* dan dicari segala kombinasi *valid* yang dapat dibentuk dari urutan posisi angka tersebut menggunakan *nested loop*.
4. Dibentuk sebuah *array* tambahan yang menampung segala kombinasi urutan posisi angka yang telah digunakan. Hal ini digunakan untuk menghindari penggunaan kombinasi urutan posisi angka yang sama secara lebih dari sekali untuk kasus adanya kartu yang bernilai sama.
5. Untuk setiap iterasi kombinasi urutan posisi angka yang valid, dilakukan iterasi sebanyak 64 kali menggunakan *nested loop* untuk jumlah kombinasi yang dapat dibentuk dari aspek operator aritmatika yang digunakan. Dalam hal ini digunakan *integer* dari 0 sampai 3 yang masing-masing secara berturut merepresentasikan operator +, -, *, dan / .
6. Dalam setiap iterasi di dalam iterasi yang diproses, akan dilakukan perhitungan sebanyak 5 kali untuk menghitung hasil dari setiap kombinasi pengurungan yang mungkin. Dalam hal ini, fungsi *calculate1* sampai *calculate5* secara berturut-turut merepresentasikan jenis pengurungan yang mungkin sesuai dengan jenis pengurungan yang telah tertera sebelumnya.
7. Setelah program selesai melakukan perhitungan dan menampilkan jumlah solusi serta semua operasi solusi yang mungkin, program akan kembali meminta *input* dari user mengenai apakah solusi tersebut ingin disimpan dalam sebuah *text file*.
8. Jika *user* ingin menyimpan solusi tersebut dalam *text file*, maka program akan meminta *user* untuk memasukkan nama *file* tempat solusi akan disimpan dan menyimpan solusi pada *file* sesuai nama yang dimasukkan. Di sisi lain, jika *user* tidak menginginkannya, maka program akan selesai dijalankan.

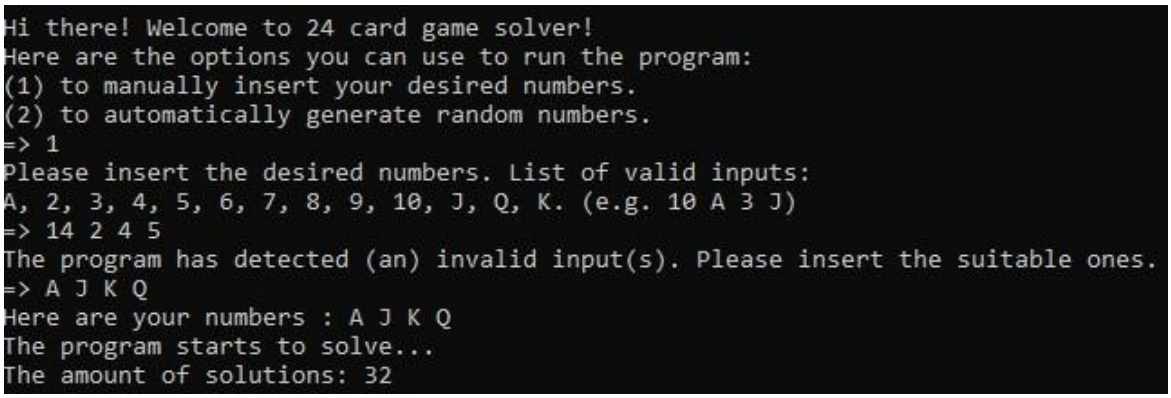
Bab IV
Implementasi Program

A. *Screenshot* Pelaksanaan Program

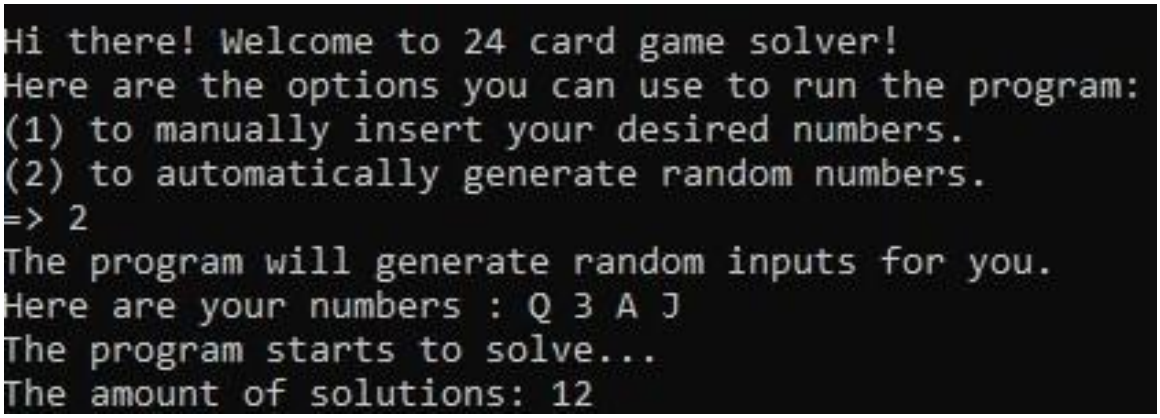
Di bawah ini, dipaparkan kumpulan *screenshot* ketika program dijalankan secara umum.



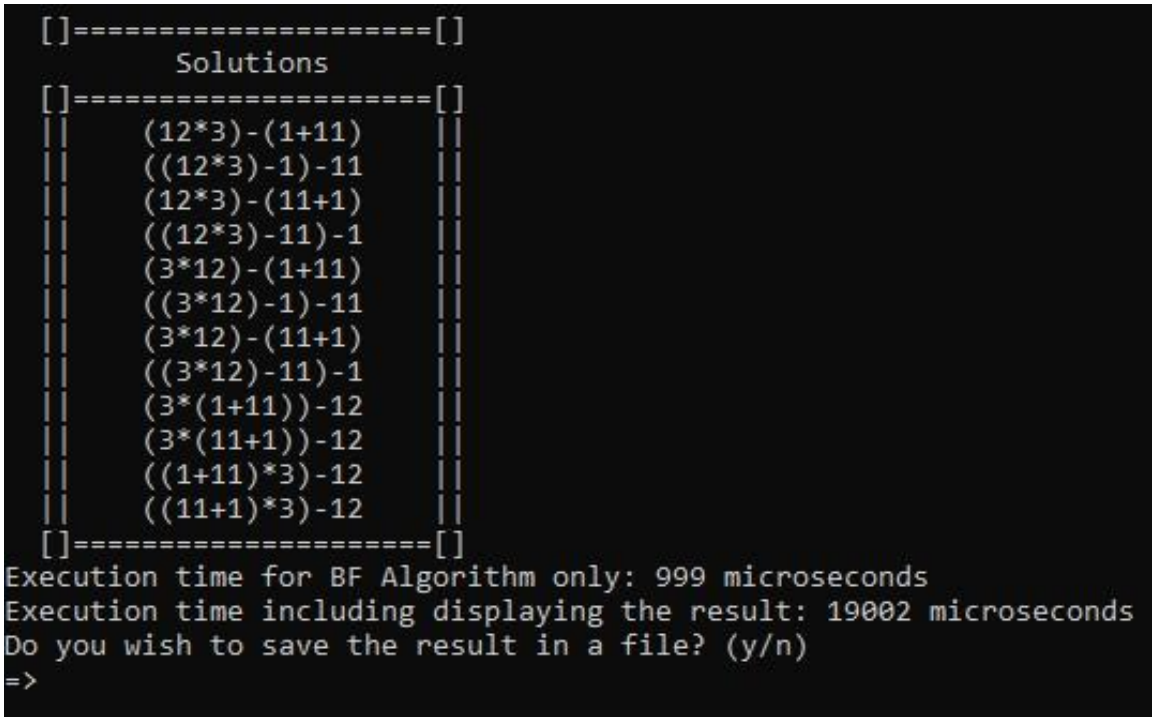
Gambar 4.1 *Screenshot* saat Program Dijalankan



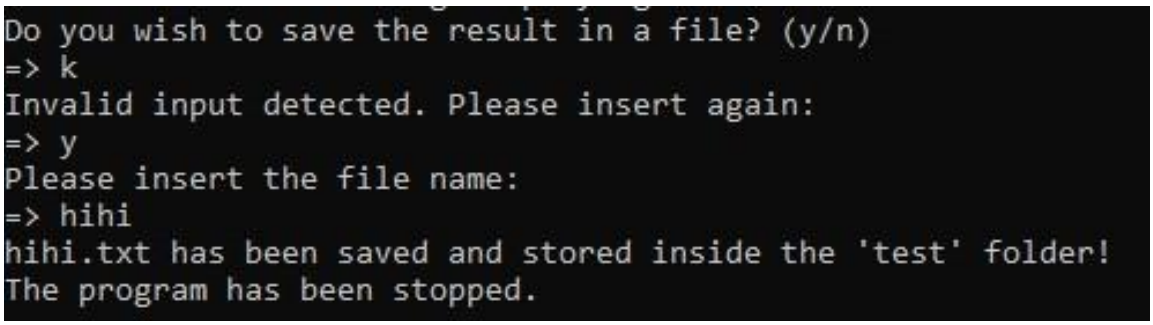
Gambar 4.2 *Screenshot* saat User Memilih Pilihan Pertama



Gambar 4.3 *Screenshot* saat User Memilih Pilihan Kedua



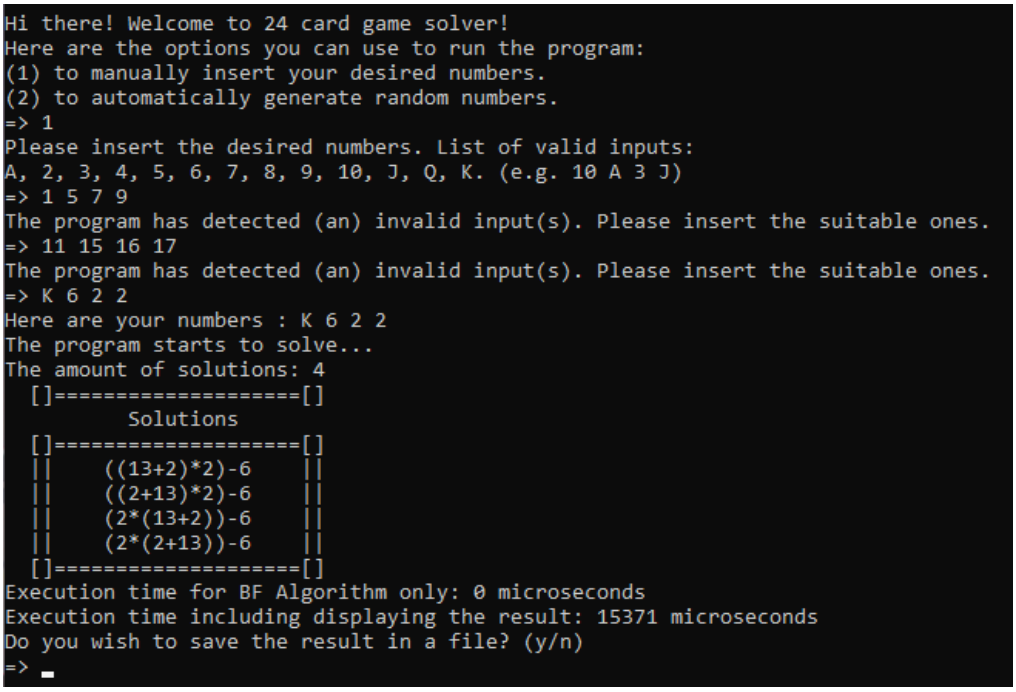
Gambar 4.4 *Screenshot* saat Program Menampilkan Semua Solusi yang Valid



Gambar 4.5 *Screenshot* saat Solusi akan Disimpan pada Text File

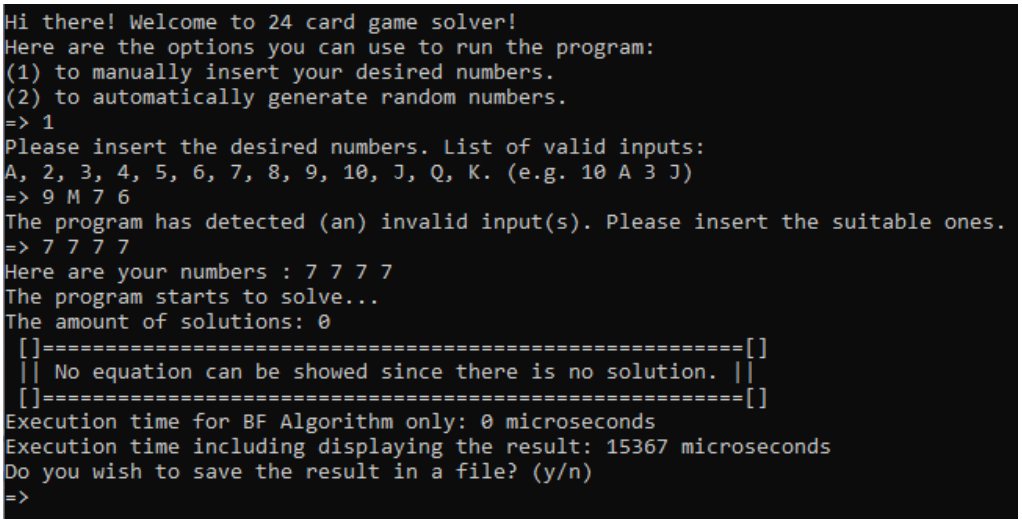
Di bawah ini, ditampilkan *screenshot* pada aspek *input* dan *output* program ketika program dijalankan.

1. *User* melakukan *input* secara manual dan permasalahan memiliki solusi



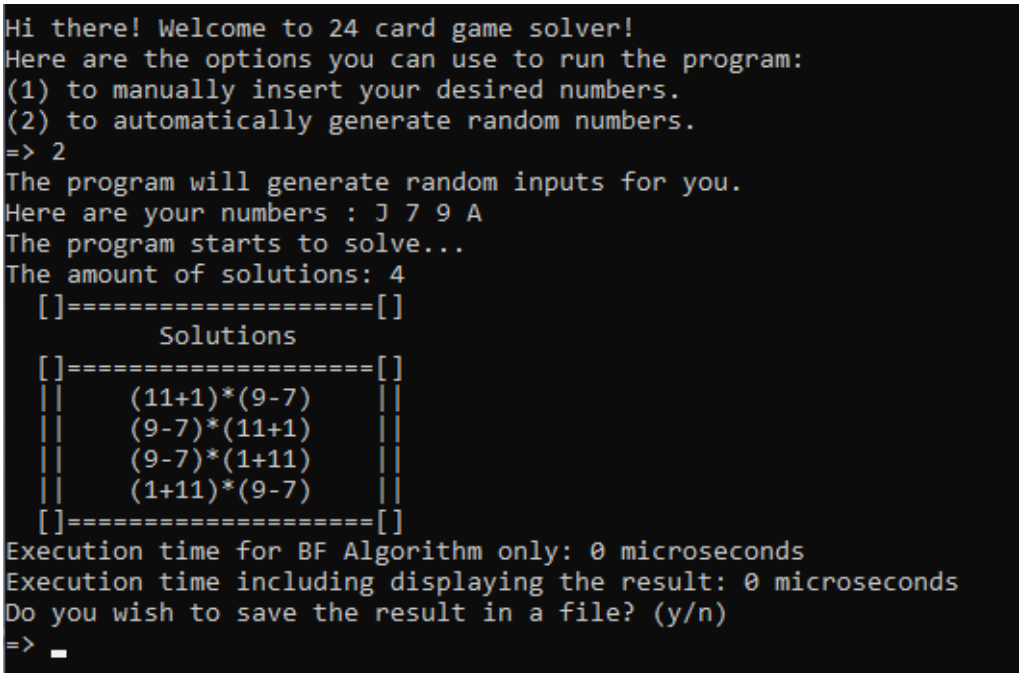
Gambar 4.6 Kasus *input* manual dan permasalahan memiliki solusi

2. *User* melakukan *input* secara manual dan permasalahan tidak memiliki solusi



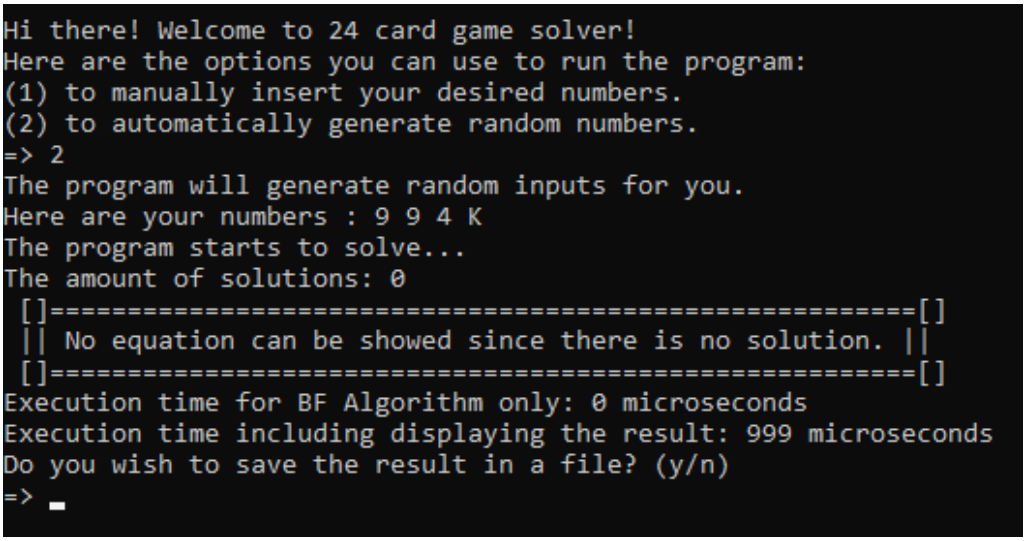
Gambar 4.7 Kasus *input* manual dan permasalahan tidak memiliki solusi

3. *User* meminta *input* otomatis dan permasalahan memiliki solusi



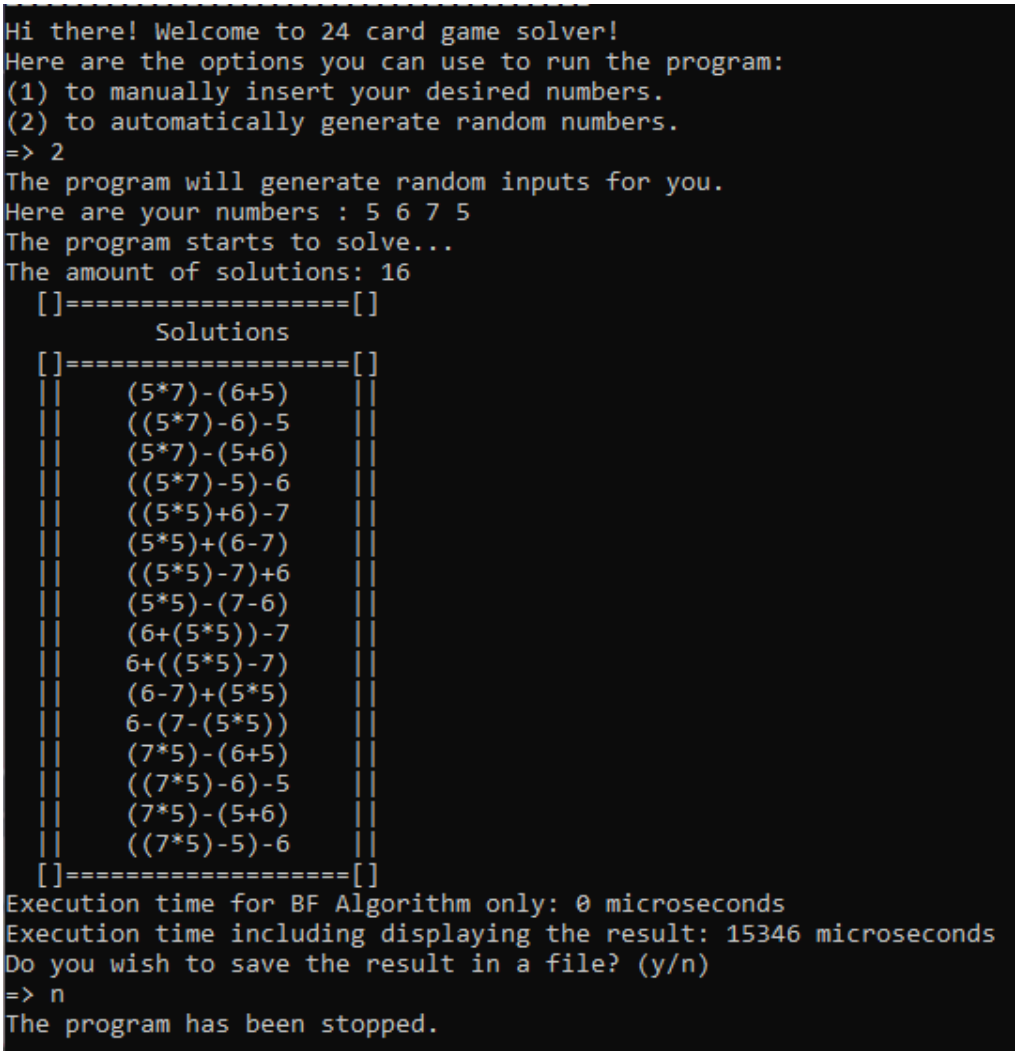
Gambar 4.8 Kasus *input* otomatis dan permasalahan memiliki solusi

4. *User* meminta *input* otomatis dan permasalahan tidak memiliki solusi



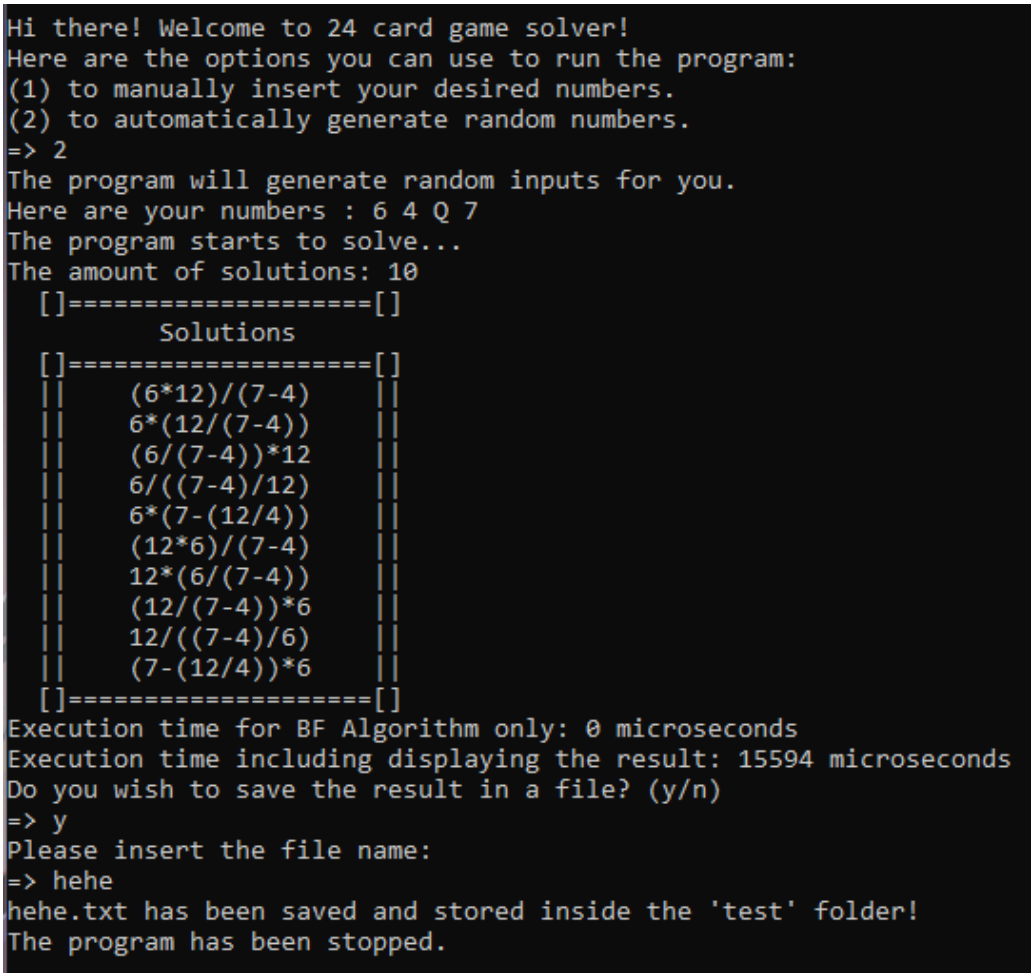
Gambar 4.9 Kasus *input* otomatis dan permasalahan tidak memiliki solusi

5. *User* tidak ingin menyimpan solusi pada *text file*

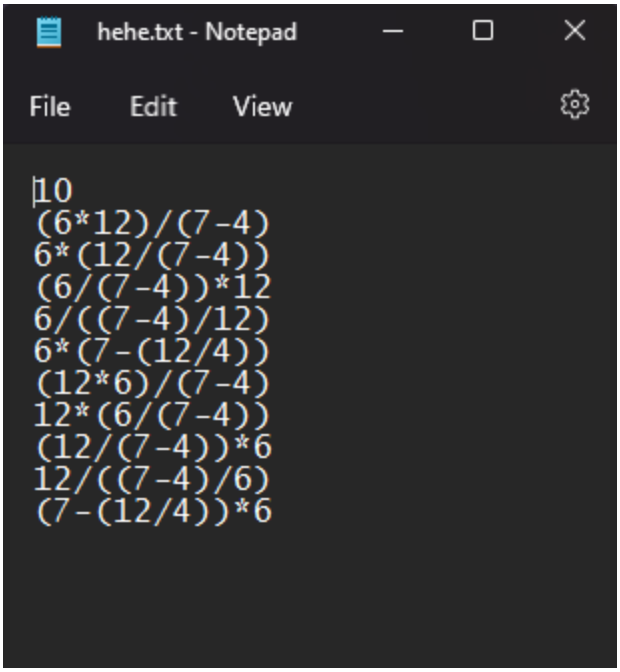


Gambar 4.10 Kasus *user* tidak ingin menyimpan solusi pada *text file*

6. *User* ingin menyimpan solusi pada *text file*



Gambar 4.11 Kasus *user* ingin menyimpan solusi pada *text file*



Gambar 4.12 Isi *file hehe.txt* tempat solusi tersebut disimpan

B. Rekapitulasi Percobaan

Berikut adalah pemaparan *sample* hasil percobaan yang dilakukan sebanyak 20 kali menggunakan *randomizer* yang disediakan oleh program. Tabel terdiri dari 4 kolom, yaitu percobaan ke-n, kombinasi kartu yang digunakan, banyak solusi yang ada, dan waktu yang dibutuhkan untuk mengeksekusi program sampai semua solusi dipaparkan.

Tabel 4.1 Tabel Pemaparan Hasil Percobaan

No	Kombinasi Kartu	Banyak Solusi	Waktu Eksekusi (µs)
1	3 Q 7 8	116	108001
2	8 8 A 2	66	70002
3	J 9 4 8	110	100002
4	6 9 2 8	94	92000
5	3 2 K 9	18	34004
6	9 Q K J	16	14004
7	K 6 2 2	4	2999

8	9 10 8 4	16	14001
9	7 2 4 2	24	23002
10	J 9 A 6	0	999
11	J 5 3 6	26	25001
12	J J 2 5	0	1000
13	2 3 A 4	242	231003
14	2 10 2 9	6	5999
15	3 Q 3 9	20	22000
16	6 K J 3	16	13000
17	10 7 2 3	20	18000
18	A 9 6 2	72	65998
19	9 10 A J	0	999
20	2 6 9 3	80	91002

Bab V
Penutup

Permainan 24 adalah permainan kartu yang berdasar pada penggunaan operasi aritmatika dalam memanipulasi 4 nilai pada kartu menjadi angka 24. Cara dan perhitungan yang dilakukan untuk mendapat nilai 24 tentunya bergantung pada kecerdasan dan kreativitas sang pemain dalam melihat 4 nilai yang dimilikinya. Namun, tanpa memikirkan efisiensi dan efektivitas penemuan solusi, pencarian penyelesaian masalah dengan mencoba satu per satu segala kemungkinan yang ada bisa saja dilakukan. Dengan begitu, penyelesaian permainan 24 dapat ditentukan dengan menggunakan algoritma *brute force*.

Dalam menggunakan algoritma *brute force*, terdapat 3 aspek yang perlu diperhatikan, yaitu operator aritmatika yang digunakan, jenis pengurungan, dan urutan posisi angka. Terdapat 64 kombinasi yang dapat dibentuk dari posisi dan jenis operator aritmatika yang digunakan, 5 kombinasi dari jenis pengurungan, dan 24 kombinasi dari urutan posisi angka. Dengan begitu, jumlah kombinasi maksimal yang dibentuk program adalah 7680 kombinasi. Meski begitu, tidak dapat dipungkiri bahwa terdapat juga kemungkinan bahwa 4 nilai kartu yang dimiliki memang mustahil untuk membentuk nilai 24 setelah perhitungan untuk segala kombinasi yang mungkin.

Daftar Pustaka

GeeksforGeeks. (n.d.-a). *Getline (string) in C++*. Retrieved January 18, 2023, from https://www.geeksforgeeks.org/getline-string-c/

GeeksforGeeks. (n.d.-b). *Measure execution time of a function in C++*. Retrieved January 20, 2023, from https://www.geeksforgeeks.org/measure-execution-time-function-cpp/

GeeksforGeeks. (n.d.-c). *Rand() and srand() in C++*. Retrieved January 18, 2023, from https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/#:~:text=Save%20Article-,rand(),a%20series%20of%20random%20numbers.

Mulani, S. (2022, August 4). *String Concatenation in C++: 4 Ways to Concatenate Strings*. https://www.digitalocean.com/community/tutorials/string-concatenation-in-c-plus-plus

Munir, R. (n.d.). *Algoritam Brute Force (Bagian 1)*. Retrieved January 23, 2023, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

Ravikiran, A. S. (2022, December 26). *Convert Int to String in C++ Using Different Methods*. https://www.simplilearn.com/tutorials/cpp-tutorial/int-to-string-cpp

w3schools. (n.d.-d). *C++ Files*. Retrieved January 21, 2023, from https://www.w3schools.com/cpp/cpp_files.asp

Lampiran

A. Tabel *Progress Checklist*

No	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	✓	

B. Github *Repository Link*
https://github.com/Gulilil/Tucil1_13521116