**Laporan Tugas Kecil 1 IF2211 Strategi Algoritma**

# PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA BRUTE FORCE


Disusun oleh:

Naufal Syifa Firdaus 13521050

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG 2023**

# BAB I: PENDAHULUAN

## A. Permainan Kartu 24

24 adalah salah satu dari sekian banyak permainan yang menggunakan set kartu remi. Permainan 24 dimainkan dengan cara pemain mengambil 4 kartu dari tumpukan kartu yang telah diacak. Pemain lalu berusaha menghitung dan menaksir ekspresi matematika dengan kombinasi angka dari 4 kartu tersebut dan juga operasi dasar +,-,*, dan / yang menghasilkan nilai 24. Pemain dapat mengkombinasikan angka dan operasi tersebut sebebas mungkin. Kartu huruf dengan yaitu A, J, Q, dan K secara berurutan bernilai 1, 11, 12, dan 13. Simbol sekop, hati, wajik, dan keriting tidak berpengaruh apapun terhadap permainan. Sebagai contoh, pemain mengambil kartu A, J, 9, dan 8 dengan begitu salah satu solusinya adalah 8 * (A - (9 - J)) karena ekspresi tersebut jika dihitung hasilnya adalah 24.

Sekilas permainan tersebut adalah adu kekuatan dan kecepatan menghitung otak manusia, tapi permainan tersebut dapat dengan mudah diselesaikan dengan menggunakan konsep permutasi. Tentunya kemungkinan permutasi dari persoalan diatas tidak sedikit, maka dari itu, dapat digunakan bantuan komputer untuk menyelesaikannya, lebih tepatnya lagi menggunakan algoritma *Brute Force*.

## B. Algoritma *Brute Force*

*Brute Force* sesuai namanya adalah algoritma yang menyelesaikan masalah dengan cara langsung dan memanfaatkan kapabilitas komputasi dari komputer. Algoritma ini seringkali diterapkan dengan cara mengiterasi semua kemungkinan pemecahan masalah dan menghasilkan solusi ketika ditemukan. *Brute Force* tidak mengindahkan efisiensi waktu dan pengaplikasiannya cenderung tidak terlalu kompleks. Untuk mencari solusi dari permainan 24, algoritma *Brute Force* dapat diterapkan dengan mengiterasi semua kemungkinan ekspresi dari 4 angka kartu dan operatornya.

# BAB II: PENERAPAN ALGORITMA BRUTE FORCE

A. Banyak Kemungkinan

Sebelum menerapkan *Brute Force,* pertama harus diketahui dahulu banyaknya kemungkinan yang mungkin ada dalam gabungan ekspresi yang dapat dibentuk oleh 4 angka dan 3 operator. Hal yang perlu diperhatikan adalah banyaknya kemungkinan urutan angka, urutan operator dan bentuk ekspresi.

Dimulai dari urutan angka, ada berapa banyak cara untuk mengurutkan 4 angka yang berbeda? Menggunakan konsep permutasi, banyaknya cara dapat dihitung menggunakan faktorial, $4! = 4 \times 3 \times 2 \times 1 = 24$. Maka banyaknya cara mengurutkan 4 angka ada 24 cara.

1, 2, 3, 4
1, 2, 4, 3
1, 3, 2, 4
(dst.)
4, 3, 2, 1

Lalu berapa banyak kemungkinan cara menempatkan operator dalam sebuah ekspresi? Permasalahan tersebut dapat dianalogikan seperti tiga tempat yang masing-masing dapat diisi satu bola. Bola yang dapat dimasukan ada 4 jenis yaitu +, -, *, dan /. Terlihat bahwa permasalahan tersebut dapat diselesaikan dengan mengalikan banyak jenis bola sebanyak tiga kali, $4 \times 4 \times 4 = 64$. Maka banyaknya kombinasi operator dalam ekspresi ada 64.

+, +, +
+, +, -
(dst.)
/, /, /

Terakhir, karena jumlahnya sedikit, bentuk ekspresi dapat ditentukan secara manual yaitu:

((a * b) * c) * d
(a * (b * c)) * d
(a * b) * (c * d)
a * ((b * c) * d)
a * (b * (c * d))

Sehingga terdapat 5 bentuk ekspresi yang mungkin.

Dengan memperhitungkan semua kemungkinan di atas total ekspresi yang harus dievaluasi secara *Brute Force* dapat dihitung dengan mengalikan semua hasil diatas, $24 \times 64 \times 5 = 7680$. Maka dari itu, total semua ekspresi yang mungkin ada 7680 ekspresi matematika. Keseluruhan ekspresi tersebut harus dihitung dan dicek hasilnya.

B. Algoritma

*Brute Force* dapat mengiterasi 7680 ekspresi secara bertahap. Pertama, tinjau semua kemungkinan gabungan operator yang jumlahnya ada 64. Pada setiap iterasi gabungan operator, tinjau semua kemungkinan urutan angka yang ada 24 buah. Pada setiap iterasi urutan angka yang ada di setiap iterasi gabungan operator, tinjau semua kemungkinan bentuk ekspresi. Gabungan dari operator, urutan angka, dan bentuk ekspresi akan membentuk sebuah ekspresi yang lengkap sekaligus dapat dihitung hasilnya.

C. Penerapan Algoritma sebagai Sebuah Program

Dalam sebuah program, algoritma yang dimengerti oleh manusia harus diterjemahkan kedalam algoritma yang bisa diaplikasikan pada bahasa pemrograman. Berikut adalah langkah-langkah yang dapat diterapkan pada sebuah bahasa pemrograman dengan berdasarkan pada algoritma di atas yang sudah didefinisikan.

1. Definisikan setiap 24 kemungkinan urutan angka dengan sebuah fungsi yang dapat dipanggil. Fungsi tersebut akan menghasilkan array urutan angka sesuai argumen yang diberikan. Argumen pada fungsi adalah urutan angka awal dan integer yang menunjukan permutasi keberapa yang diminta pemanggil fungsi.
2. Untuk mengiterasi 64 kemungkinan operator, buat 3 *loop* dengan *loop* pertama berisi *loop* kedua dan *loop* kedua berisi *loop* ketiga. Setiap loopnya mengiterasi 4 kemungkinan operator.
3. Pada *loop* ketiga, buat *loop* yang mengiterasi 24 kemungkinan urutan angka dengan memanggil fungsi terkait dengan argumen 0 sampai 23.
4. Dengan urutan angka dan 3 operator yang sudah terdefinisi, buat lima kondisional yang setiap kondisinya adalah gabungan operator, angka, dan bentuk ekspresi. Kondisi terpenuhi jika ekspresi yang terbentuk menghasilkan nilai 24. Ekspresi yang memenuhi dapat disimpan kedalam sebuah array of string untuk ditampilkan kemudian.

# BAB III: PROGRAM DALAM C++

Berikut adalah source code program dalam bahasa c++ yang dikategorikan berdasarkan file dan fungsi.

A. main.cpp

| | | |
|---|---|---|
| main() | Argumen: -<br>Return Value: integer | Program utama yang menjalankan semua fungsi dan program dari awal sampai akhir. |

```
int main (){
    double * cardValue;
    double * permValue;
    string resultExpression = "";
    const int goal = 24;
    int resultCount = 0;
    double resultTime;
    string resultArray[7680];

    printHomeScreen();

    cardValue = input();

    while (!inputValid(cardValue)){
        cout << " _____________________________________________________________________________ \n";
        cout << "|                            Invalid Input!                                   | \n";
        cout << "| Input four of the following = A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K  |\n";
        cout << "|_____________________________________________________________________________|\n";
        cardValue = input();
    }

    cout << " _____________________________________________________________________________ \n";
    cout << "|                         Your Cards Value:                                   | \n";
    cout << "|                             ";
    for (int i = 0; i < 4 ;i++){

        cout << cardValue[i] << " ";
    }
    cout << "                                    | \n";
    cout << "|_____________________________________________________________________________|\n";
    cout << "\n";
```

```
    auto timeBefore = high_resolution_clock::now();
    solve(cardValue, goal, &resultCount, resultArray);
    auto timeAfter = high_resolution_clock::now();
    auto timeTotal = duration_cast<microseconds>(timeAfter - timeBefore);

    resultTime = timeTotal.count();

    cout << "RESULT COUNT: " << resultCount << "\n";
    if (resultCount != 0){

        cout << "PROCESS DURATION: " << resultTime << " microseconds \n";
        //cout << resultExpression;


        cout << "\n" << "Result Expression: ";
        for(int i = 0; i < resultCount; i++){
            if (i % 3 ==0){
                cout << "\n";
            }
            cout << resultArray[i] << "       ";
        }
        cout << "\n \n";
        cout << " _____________________________________________________________________________ \n";
        cout << "|                Do you want to store the result? (y/n)                       |\n";
        cout << "|_____________________________________________________________________________|\n";
        char opt;
        cin >> opt;
        if (opt == 'y'){
            storeInFile(resultArray, resultCount, resultTime, cardValue);
        }
        else {
            cout << " _____________________________________________________________________________ \n";
            cout << "|                Result not saved. Thankyou and goodbye!                      |\n";
            cout << "|_____________________________________________________________________________|\n";
        }
    }
    else{
        cout << " _____________________________________________________________________________ \n";
        cout << "|              There is no solution for this set of cards.                    |\n";
        cout << "|_____________________________________________________________________________|\n";
    }


    return 0;
}
```

B. toolsFunction.cpp

| printHomeScreen() | Argumen: -<br>Return Value: - | Menampilkan informasi umum ke layar beserta splash screen dan panduan menggunakan program |
|---|---|---|

```
void printHomeScreen(){
// Menampilkan informasi dan pesan selamat datang pada pengguna

    cout << " ____________________________________________________________________________ \n";
    cout << "|                                                                            |\n";
    cout << "|                            24 CARD GAME SOLVER                             |\n";
    cout << "|                                                                            |\n";
    cout << "|               Input 4 card in a certain order and the                      |\n";
    cout << "|               program will determine all expression                        |\n";
    cout << "|               that result in 24 total value.                               |\n";
    cout << "|                                                                            |\n";
    cout << "|                         Input Example: A 10 2 K                            |\n";
    cout << "|                                                                            |\n";
    cout << "|     accepted card input = A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K           |\n";
    cout << "|                                                                            |\n";
    cout << "|                                          Created By: Naufal Syifa F.       |\n";
    cout << "|____________________________________________________________________________|\n";

}
```

| inputValid() | Argumen: double * val<br>Return Value: boolean | Menghasilkan true jika val tidak -1 yaitu adalah nilai kartu yang diterima. |
|---|---|---|

```
bool inputValid(double * val){
// Menghasilkan true jika masukan adalah valid

    bool valid = true;

    for(int i = 0; i < 4; i++){
        if (val[i] == -1){
            valid = false;
```

```
        }
    }
    return valid;
}
```

| valueOfCard() | Argumen: string card<br>Return Value: double | Menerima satu string yang merepresentasikan kartu dan menghasilkan nilai kartu tersebut. |
|---|---|---|

```
double valueOfCard(string card){
// Menghasilkan nilai dari kartu yang dimasukan
// Kalau kartu tidak valid nilai akan menjadi -1

    double val;

    if (card ==  "A"){
        val = 1;
    }
    else if (card == "2"){
        val = 2;
    }
    else if (card == "3"){
        val = 3;
    }
    else if (card == "4"){
        val = 4;
    }
    else if (card == "5"){
        val = 5;
    }
    else if (card == "6"){
        val = 6;
    }
    else if (card == "7"){
        val = 7;
    }
    else if (card == "8"){
        val = 8;
```

```
    }
    else if (card == "9"){
        val = 9;
    }
    else if (card == "10"){
        val = 10;
    }
    else if (card == "J"){
        val = 11;
    }
    else if (card == "Q" ){
        val = 12;
    }
    else if (card == "K"){
        val = 13;
    }
    else {
        val = -1;
    }

    return val;
}
```

| | | |
|---|---|---|
| input() | Argumen: -<br>Return Value: double * | Menampilkan pilihan untuk input (random atau dari pengguna), menerima input pengguna, menghasilkan input random, dan menghasilkan array of double yang adalah urutan 4 kartu. |

```
double * input(){
// Menerima masukan kartu dari pengguna atau memasukan 4 kartu random
// Lalu fungsi menghasilkan array of double yang berisi 4 nilai kartu

    string card[4];
    static double cardValue[4];
    char opt;

    cout << " _____________________________________________________________________ \n";
    cout << "|               Do you want to randomize the input? (y/n)             | \n";
    cout << "|_____________________________________________________________________|\n";
```

```
    cin >> opt;

    if( opt == 'n'){
        cout << "Card Input: ";
        for (int i = 0; i < 4; i++){
            cin >> card[i];
            //cout << card[i] << "\n";
            cardValue[i] = valueOfCard(card[i]);
    }
    }
    else{
        srand(time(0)); // set up random seed
        for (int i = 0; i < 4 ;i++){
            cardValue[i] = rand() % 13 + 1; // random number generator
        }
    }


    return cardValue;
}
```

| | | |
|---|---|---|
| permutation() | Argumen: double * , integer<br>Return Value: double * | Menghasilkan permutasi urutan angka pada 4 kartu sesuai dengan permutasi keberapa yang diminta. |

```
double * permutation(double * cardValue, int i){
// Menghasilkan hasil permutasi urutan kartu sesuai urutan yang dimasukan (1-24) karena ada
// 24 kemungkinan susunan angka.

    static double permValue[4];

    switch (i)
    {
    case 0:
        permValue[0] = cardValue[0]; permValue[1] = cardValue[1]; permValue[2] = cardValue[2]; permValue[3] = cardValue[3];
        break;
    case 1:
        permValue[0] = cardValue[0]; permValue[1] = cardValue[1]; permValue[2] = cardValue[3]; permValue[3] = cardValue[2];
        break;
    case 2:
        permValue[0] = cardValue[0]; permValue[1] = cardValue[2]; permValue[2] = cardValue[1]; permValue[3] = cardValue[3];
        break;
    case 3:
        permValue[0] = cardValue[0]; permValue[1] = cardValue[2]; permValue[2] = cardValue[3]; permValue[3] = cardValue[1];
        break;
    case 4:
        permValue[0] = cardValue[0]; permValue[1] = cardValue[3]; permValue[2] = cardValue[1]; permValue[3] = cardValue[2];
        break;
    case 5:
        permValue[0] = cardValue[0]; permValue[1] = cardValue[3]; permValue[2] = cardValue[2]; permValue[3] = cardValue[1];
```

```
        break;
    case 6:
        permValue[0] = cardValue[1]; permValue[1] = cardValue[0]; permValue[2] = cardValue[2]; permValue[3] = cardValue[3];
        break;
    case 7:
        permValue[0] = cardValue[1]; permValue[1] = cardValue[0]; permValue[2] = cardValue[3]; permValue[3] = cardValue[2];
        break;
    case 8:
        permValue[0] = cardValue[1]; permValue[1] = cardValue[2]; permValue[2] = cardValue[0]; permValue[3] = cardValue[3];
        break;
    case 9:
        permValue[0] = cardValue[1]; permValue[1] = cardValue[2]; permValue[2] = cardValue[3]; permValue[3] = cardValue[0];
        break;
    case 10:
        permValue[0] = cardValue[1]; permValue[1] = cardValue[3]; permValue[2] = cardValue[0]; permValue[3] = cardValue[2];
        break;
    case 11:
        permValue[0] = cardValue[1]; permValue[1] = cardValue[3]; permValue[2] = cardValue[2]; permValue[3] = cardValue[0];
        break;
    case 12:
        permValue[0] = cardValue[2]; permValue[1] = cardValue[0]; permValue[2] = cardValue[1]; permValue[3] = cardValue[3];
        break;
    case 13:
        permValue[0] = cardValue[2]; permValue[1] = cardValue[0]; permValue[2] = cardValue[3]; permValue[3] = cardValue[1];
        break;
    case 14:
        permValue[0] = cardValue[2]; permValue[1] = cardValue[1]; permValue[2] = cardValue[0]; permValue[3] = cardValue[3];
        break;
    case 15:
        permValue[0] = cardValue[2]; permValue[1] = cardValue[1]; permValue[2] = cardValue[3]; permValue[3] = cardValue[0];
        break;
    case 16:
        permValue[0] = cardValue[2]; permValue[1] = cardValue[3]; permValue[2] = cardValue[0]; permValue[3] = cardValue[1];
        break;
    case 17:
        permValue[0] = cardValue[2]; permValue[1] = cardValue[3]; permValue[2] = cardValue[1]; permValue[3] = cardValue[0];
        break;
    case 18:
        permValue[0] = cardValue[3]; permValue[1] = cardValue[0]; permValue[2] = cardValue[1]; permValue[3] = cardValue[2];
        break;
    case 19:
        permValue[0] = cardValue[3]; permValue[1] = cardValue[0]; permValue[2] = cardValue[2]; permValue[3] = cardValue[1];
        break;
    case 20:
        permValue[0] = cardValue[3]; permValue[1] = cardValue[1]; permValue[2] = cardValue[0]; permValue[3] = cardValue[2];
        break;
    case 21:
        permValue[0] = cardValue[3]; permValue[1] = cardValue[1]; permValue[2] = cardValue[2]; permValue[3] = cardValue[0];
        break;
    case 22:
        permValue[0] = cardValue[3]; permValue[1] = cardValue[2]; permValue[2] = cardValue[0]; permValue[3] = cardValue[1];
        break;
    case 23:
        permValue[0] = cardValue[3]; permValue[1] = cardValue[2]; permValue[2] = cardValue[1]; permValue[3] = cardValue[0];
        break;

    default:
        permValue[0] = cardValue[0]; permValue[1] = cardValue[1]; permValue[2] = cardValue[2]; permValue[3] = cardValue[3];
        break;
    }

    return permValue;
}
```

| | | |
|---|---|---|
| ex() | Argumen: double, char<br>Return Value: double | Menghasilkan hasil operasi antara dua double argumen sesuai operator yang diminta. |

```
double ex(double a,char oper, double b){
// Menghasilkan sebuah double hasil dari perhitungan nilai a dan b
// sesuai dengan operasi yang dimasukan

    double res;

    switch (oper)
    {
    case '+':
        res = a + b;
        break;
    case '-':
        res = a - b;
        break;
    case '/':
        res = a / b;
        break;
    case '*':
        res = a * b;
        break;

    default:
        break;
    }

    return res;
}
```

| | | |
|---|---|---|
| convertCharToString() | Argumen: char *, int<br>Return Value: string | Menghasilkan string yang isinya adalah tiap elemen dari array of char dari argumen. |

```
string convertCharToString(char * a, int n){
// Menghasilkan string dari array of char

    string res = "";

    for(int i = 0; i < n; i++ ){
        if (a[i] != ' '){
            res += a[i];
         }
    }


    return res;
}
```

| | | |
|---|---|---|
| solve() | Argumen: double *, double, int *, string *<br>Return Value: - | Menerima urutan kartu, nilai target, jumlah hasil, dan array of string. Fungsi mengiterasi semua kemungkinan dengan algoritma Brute Force dan menyimpan hasil yang valid pada array of string dan jumlahnya pada integer keluaran. |

```
void solve(double * cardValue, double goal, int * count, string * resultArray){
// algoritma utama  brute force

    char opr[4] = {'+', '-', '*', '/'};
    double * permValue;
    char buffer[30];
    int n;


    for (char x : opr){
        for (char y : opr){
            for (char z : opr){
                for (int i = 0; i < 24 ; i++){
```

```
                    permValue = permutation(cardValue, i);

                    if(ex(ex(ex(permValue[0], x, permValue[1]), y,
permValue[2]),z,permValue[3]) == goal){ // ((a * b) * c) * d
                        *count = *count + 1;
                        n = sprintf(buffer, "((%.0lf %c %.0lf) %c %.0lf) %c %.0lf",
permValue[0], x, permValue[1], y, permValue[2], z, permValue[3]);
                        resultArray[*count - 1] = buffer;
                    }

if(ex(ex(permValue[0],x,ex(permValue[1],y,permValue[2])),z,permValue[3]) == goal){ // (a *
(b * c)) * d
                        *count = *count + 1;
                        n = sprintf(buffer, "(%.0lf %c (%.0lf %c %.0lf)) %c %.0lf",
permValue[0], x, permValue[1], y, permValue[2], z, permValue[3]);
                        resultArray[*count - 1] = buffer;
                    }
                    if(ex(ex(permValue[0],x,permValue[1]), y
,ex(permValue[2],z,permValue[3])) == goal){// (a * b) * (c * d)
                        *count = *count + 1;
                        n = sprintf(buffer, "(%.0lf %c %.0lf) %c (%.0lf %c %.0lf)",
permValue[0], x, permValue[1], y, permValue[2], z, permValue[3]);
                        resultArray[*count - 1] = buffer;
                    }
                    if(ex(permValue[0], x, ex(ex(permValue[1], y, permValue[2]),z
,permValue[3])) == goal){ // a * ((b * c) * d)
                        *count = *count + 1;
                        n = sprintf(buffer, "%.0lf %c ((%.0lf %c %.0lf) %c %.0lf)",
permValue[0], x, permValue[1], y, permValue[2], z, permValue[3]);
                        resultArray[*count - 1] = buffer;
                    }
                    if(ex(permValue[0], x, ex(permValue[1], y, ex(permValue[2],z
,permValue[3]))) == goal){ // a * (b * (c * d))
                        *count = *count + 1;
                        n = sprintf(buffer, "%.0lf %c (%.0lf %c (%.0lf %c %.0lf))",
permValue[0], x, permValue[1], y, permValue[2], z, permValue[3]);
                        resultArray[*count - 1] = buffer;
                    }

                }
            }
        }
    }
```

```
}
```

| | | |
|---|---|---|
| fileName() | Argumen: double *<br>Return Value: string | Menerima urutan kartu dan menghasilkan string nama file untuk menyimpan hasil komputasi. |

```
string fileName(double * cardValue){
// menghasilkan nama file sesuai format yang telah ditentukan

    string name = "Result_(";
    char card[1];

    for (int i = 0; i < 4; i++){

        if( i != 0 ){
            name += ",";
        }

        if(cardValue[i] == 1){
            name += "A";
        }
        else if(cardValue[i] == 11){
            name += "J";
        }
        else if(cardValue[i] == 12){
            name += "Q";
        }
        else if(cardValue[i] == 13){
            name += "K";
        }
        else{
            sprintf(card, "%.0lf", cardValue[i]);
            name += card;
        }
    }
    name = name + ")";
    name = "../test/" + name + ".txt";

    return name;
```

```
}
```

| | | |
|---|---|---|
| resultInFile() | Argumen: string *, int, double, double *<br>Return Value: - | Menerima hasil komputasi beserta waktu dan jumlahnya lalu membuat file dengan informasi tersebut. |

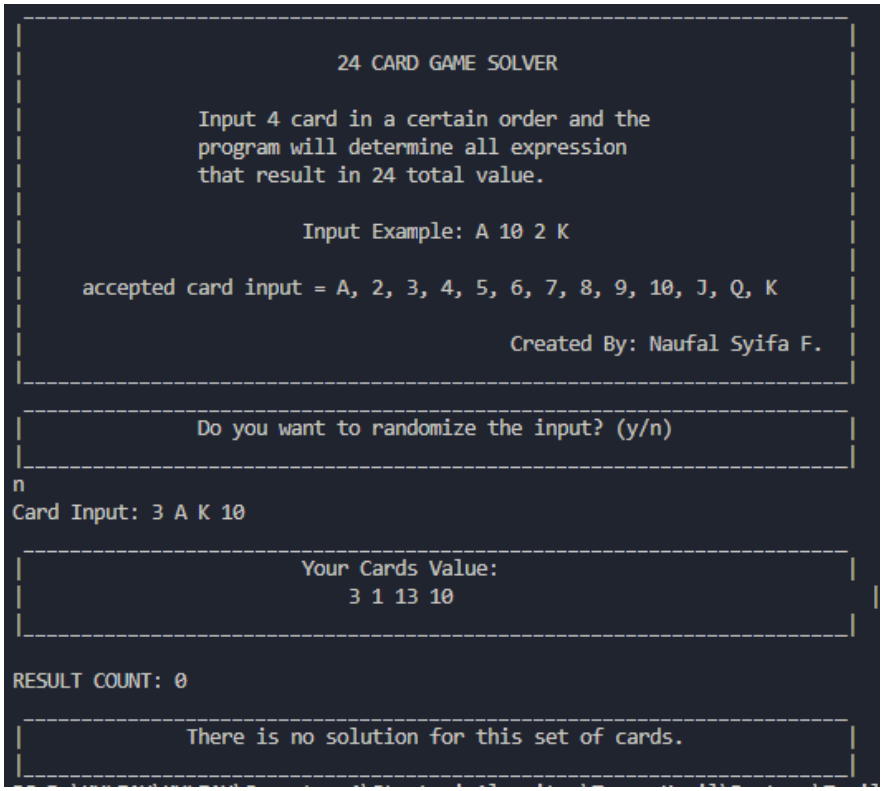
```
void storeInFile(string * resultArray, int resultCount, double resultTime, double *
cardValue){
// menyimpan hasil brute force kedalam file .txt

    string filename = fileName(cardValue);
    ofstream resultFile (filename);

    resultFile << "Card Values : " << cardValue[0] << " " << cardValue[1] << " " <<
cardValue[2] << " " << cardValue[3] << "\n";

    resultFile << resultCount << " solution found within " << resultTime << " microseconds"
<< "\n" << "\n";

    resultFile << "Result Expression: ";
    for(int i = 0; i < resultCount; i++){
        if (i % 3 ==0){
            resultFile << "\n";
        }
        resultFile << resultArray[i] << "       ";
    }

    resultFile.close();
    cout << " ____________________________________________________________________________ \n";
    cout << "|                   Result saved. Thankyou and goodbye!                    |\n";
    cout << "|____________________________________________________________________________|\n";

}
```

# BAB IV: EKSPERIMEN

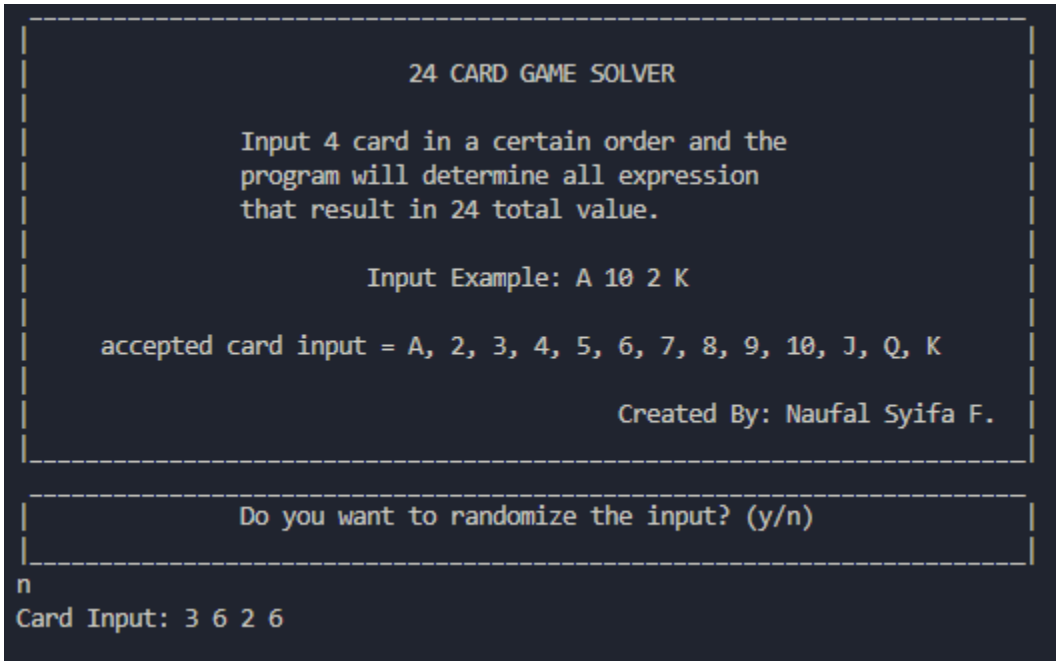
A. Nilai Kartu 7, 5, Q, 3

```
 ____________________________________________________________________
|                                                                    |
|                          24 CARD GAME SOLVER                       |
|                                                                    |
|               Input 4 card in a certain order and the              |
|               program will determine all expression                |
|               that result in 24 total value.                       |
|                                                                    |
|                        Input Example: A 10 2 K                     |
|                                                                    |
|     accepted card input = A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K   |
|                                                                    |
|                                          Created By: Naufal Syifa F.|
|____________________________________________________________________|

 ____________________________________________________________________
|               Do you want to randomize the input? (y/n)            |
|____________________________________________________________________|
y
 ____________________________________________________________________
|                        Your Cards Value:                           |
|                            7 5 12 3                                |
|____________________________________________________________________|

RESULT COUNT: 28
PROCESS DURATION: 997 microseconds

Result Expression:
((7 + 5) * 3) - 12      ((5 + 7) * 3) - 12      ((7 + 3) * 12) / 5
(7 + 3) * (12 / 5)      ((3 + 7) * 12) / 5      (3 + 7) * (12 / 5)
((7 + 3) / 5) * 12      ((3 + 7) / 5) * 12      (7 + 3) / (5 / 12)
(3 + 7) / (5 / 12)      (3 * (7 + 5)) - 12      (3 * (5 + 7)) - 12
(12 * (7 + 3)) / 5      12 * ((7 + 3) / 5)      (12 * (3 + 7)) / 5
12 * ((3 + 7) / 5)      (12 * 3) - (7 + 5)      (12 * 3) - (5 + 7)
(3 * 12) - (7 + 5)      (3 * 12) - (5 + 7)      ((12 * 3) - 7) - 5
((12 * 3) - 5) - 7      ((3 * 12) - 7) - 5      ((3 * 12) - 5) - 7
(12 / 5) * (7 + 3)      (12 / 5) * (3 + 7)      12 / (5 / (7 + 3))
12 / (5 / (3 + 7))


 ____________________________________________________________________
|               Do you want to store the result? (y/n)               |
|____________________________________________________________________|
y
 ____________________________________________________________________
|                 Result saved. Thankyou and goodbye!                |
|____________________________________________________________________|
```

B. Nilai Kartu 3, A, K, 10




C. Nilai Kartu 3, 6, 2, 6

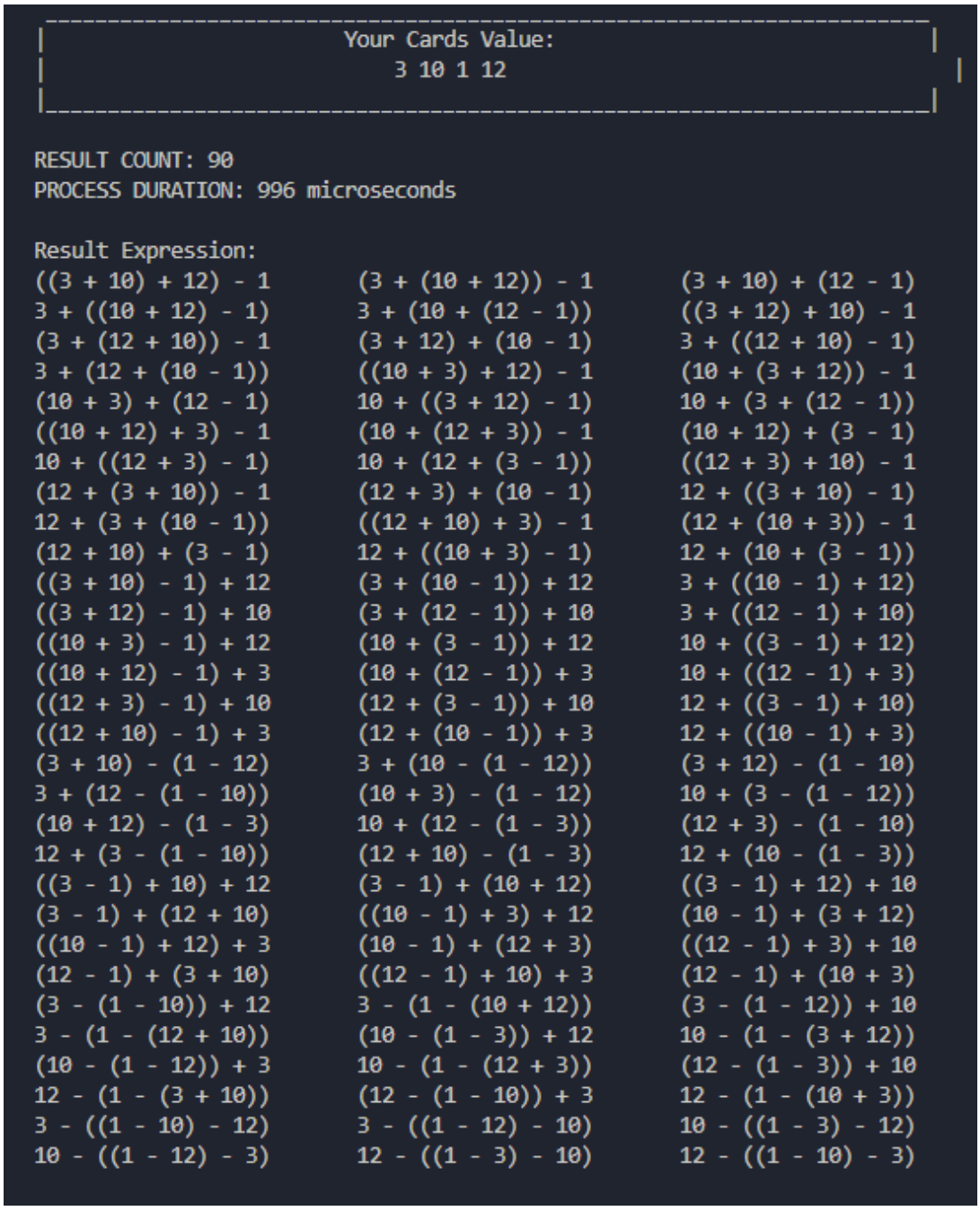


```
RESULT COUNT: 138
PROCESS DURATION: 998 microseconds

Result Expression:
6 + ((3 + 6) * 2)       6 + ((6 + 3) * 2)       6 + ((3 + 6) * 2)
6 + ((6 + 3) * 2)       ((3 + 6) * 2) + 6       ((3 + 6) * 2) + 6
((6 + 3) * 2) + 6       6 + (2 * (3 + 6))       6 + (2 * (6 + 3))
((6 + 3) * 2) + 6       6 + (2 * (3 + 6))       6 + (2 * (6 + 3))
((3 + 2) * 6) - 6       ((3 + 2) * 6) - 6       (6 + 2) * (6 - 3)
(2 + 6) * (6 - 3)       (6 + 2) * (6 - 3)       (2 + (6 / 3)) * 6
(2 + (6 / 3)) * 6       (6 - 3) * (2 + 6)       (6 - 3) * (6 + 2)
(6 - 3) * (6 + 2)       (6 - 3) * (2 + 6)       (2 * (3 + 6)) + 6
(2 * (3 + 6)) + 6       (2 * (6 + 3)) + 6       (2 * (6 + 3)) + 6
(6 * (3 + 2)) - 6       (6 * (2 + 3)) - 6       (6 * (3 + 2)) - 6
(6 * (2 + 3)) - 6       6 * (2 + (6 / 3))       6 * (2 + (6 / 3))
((3 * 6) - 6) * 2       ((3 * 6) - 6) * 2       ((6 * 3) - 6) * 2
((6 * 3) - 6) * 2       2 * ((3 * 6) - 6)       2 * ((3 * 6) - 6)
2 * ((6 * 3) - 6)       2 * ((6 * 3) - 6)       ((6 * 2) * 6) / 3
(6 * (2 * 6)) / 3       (6 * 2) * (6 / 3)       6 * ((2 * 6) / 3)
6 * (2 * (6 / 3))       ((6 * 6) * 2) / 3       (6 * (6 * 2)) / 3
(6 * 6) * (2 / 3)       6 * ((6 * 2) / 3)       6 * (6 * (2 / 3))
((2 * 6) * 6) / 3       (2 * (6 * 6)) / 3       (2 * 6) * (6 / 3)
2 * ((6 * 6) / 3)       2 * (6 * (6 / 3))       ((2 * 6) * 6) / 3
(2 * (6 * 6)) / 3       (2 * 6) * (6 / 3)       2 * ((6 * 6) / 3)
2 * (6 * (6 / 3))       ((6 * 6) * 2) / 3       (6 * (6 * 2)) / 3
(6 * 6) * (2 / 3)       6 * ((6 * 2) / 3)       6 * (6 * (2 / 3))
((6 * 2) * 6) / 3       (6 * (2 * 6)) / 3       (6 * 2) * (6 / 3)
6 * ((2 * 6) / 3)       6 * (2 * (6 / 3))       6 * ((6 / 3) + 2)
6 * ((6 / 3) + 2)       ((6 * 2) / 3) * 6       (6 * (2 / 3)) * 6
6 * ((2 / 3) * 6)       ((6 * 6) / 3) * 2       (6 * (6 / 3)) * 2
6 * ((6 / 3) * 2)       ((2 * 6) / 3) * 6       (2 * (6 / 3)) * 6
2 * ((6 / 3) * 6)       ((2 * 6) / 3) * 6       (2 * (6 / 3)) * 6
2 * ((6 / 3) * 6)       ((6 * 6) / 3) * 2       (6 * (6 / 3)) * 2
6 * ((6 / 3) * 2)       ((6 * 2) / 3) * 6       (6 * (2 / 3)) * 6
6 * ((2 / 3) * 6)       (6 * 2) / (3 / 6)       6 * (2 / (3 / 6))
(6 * 6) / (3 / 2)       6 * (6 / (3 / 2))       (2 * 6) / (3 / 6)
2 * (6 / (3 / 6))       (2 * 6) / (3 / 6)       2 * (6 / (3 / 6))
(6 * 6) / (3 / 2)       6 * (6 / (3 / 2))       (6 * 2) / (3 / 6)
6 * (2 / (3 / 6))       ((6 / 3) + 2) * 6       ((6 / 3) + 2) * 6
((6 / 3) * 2) * 6       (6 / 3) * (2 * 6)       ((6 / 3) * 6) * 2
(6 / 3) * (6 * 2)       ((2 / 3) * 6) * 6       (2 / 3) * (6 * 6)
((2 / 3) * 6) * 6       (2 / 3) * (6 * 6)       ((6 / 3) * 6) * 2
(6 / 3) * (6 * 2)       ((6 / 3) * 2) * 6       (6 / 3) * (2 * 6)
(6 / (3 / 2)) * 6       6 / (3 / (2 * 6))       (6 / (3 / 6)) * 2
6 / (3 / (6 * 2))       (2 / (3 / 6)) * 6       2 / (3 / (6 * 6))
(2 / (3 / 6)) * 6       2 / (3 / (6 * 6))       (6 / (3 / 6)) * 2
6 / (3 / (6 * 2))       (6 / (3 / 2)) * 6       6 / (3 / (2 * 6))
6 / ((3 / 2) / 6)       6 / ((3 / 6) / 2)       2 / ((3 / 6) / 6)
2 / ((3 / 6) / 6)       6 / ((3 / 6) / 2)       6 / ((3 / 2) / 6)


 ------------------------------------------------------------------------
|                Do you want to store the result? (y/n)                  |
 ------------------------------------------------------------------------
```

D. Nilai Kartu 7, 3, 2, 3

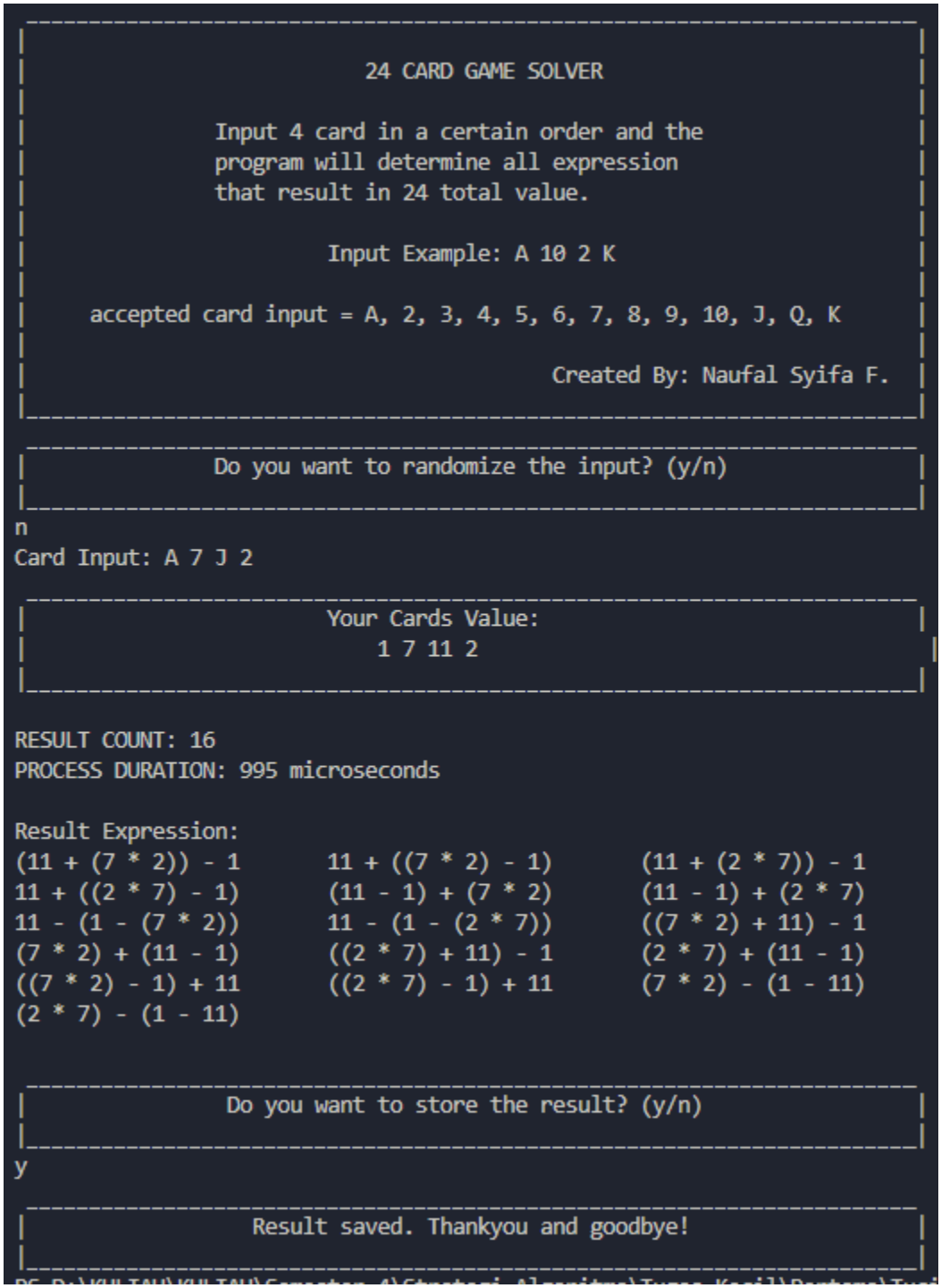
```
 ------------------------------------------------------------------
|                                                                  |
|                          24 CARD GAME SOLVER                     |
|                                                                  |
|              Input 4 card in a certain order and the             |
|              program will determine all expression               |
|              that result in 24 total value.                      |
|                                                                  |
|                        Input Example: A 10 2 K                   |
|                                                                  |
|     accepted card input = A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K |
|                                                                  |
|                                       Created By: Naufal Syifa F.|
|__________________________________________________________________|
 ------------------------------------------------------------------
|              Do you want to randomize the input? (y/n)           |
|__________________________________________________________________|
y
 ------------------------------------------------------------------
|                        Your Cards Value:                         |
|                            7 3 2 3                               |
|__________________________________________________________________|

RESULT COUNT: 64
PROCESS DURATION: 994 microseconds

Result Expression:
((7 + 3) - 2) * 3       (7 + (3 - 2)) * 3       ((7 + 3) - 2) * 3
(7 + (3 - 2)) * 3       ((3 + 7) - 2) * 3       (3 + (7 - 2)) * 3
((3 + 7) - 2) * 3       (3 + (7 - 2)) * 3       ((7 + 2) * 3) - 3
((7 + 2) * 3) - 3       ((2 + 7) * 3) - 3       ((2 + 7) * 3) - 3
((7 - 2) + 3) * 3       ((7 - 2) + 3) * 3       ((3 - 2) + 7) * 3
((3 - 2) + 7) * 3       (7 - (2 - 3)) * 3       (7 - (2 - 3)) * 3
(3 - (2 - 7)) * 3       (3 - (2 - 7)) * 3       ((7 - 3) * 2) * 3
(7 - 3) * (2 * 3)       ((7 - 3) * 3) * 2       (7 - 3) * (3 * 2)
((7 - 3) * 3) * 2       (7 - 3) * (3 * 2)       ((7 - 3) * 2) * 3
(7 - 3) * (2 * 3)       (3 * (7 + 2)) - 3       3 * ((7 + 3) - 2)
3 * (7 + (3 - 2))       (3 * (2 + 7)) - 3       3 * ((3 + 7) - 2)
3 * (3 + (7 - 2))       3 * ((7 + 3) - 2)       3 * (7 + (3 - 2))
(3 * (7 + 2)) - 3       3 * ((3 + 7) - 2)       3 * (3 + (7 - 2))
(3 * (2 + 7)) - 3       3 * ((7 - 2) + 3)       3 * ((3 - 2) + 7)
3 * ((7 - 2) + 3)       3 * ((3 - 2) + 7)       3 * (7 - (2 - 3))
3 * (3 - (2 - 7))       3 * (7 - (2 - 3))       3 * (3 - (2 - 7))
(3 * (7 - 3)) * 2       3 * ((7 - 3) * 2)       (2 * (7 - 3)) * 3
2 * ((7 - 3) * 3)       (2 * (7 - 3)) * 3       2 * ((7 - 3) * 3)
(3 * (7 - 3)) * 2       3 * ((7 - 3) * 2)       (3 * 2) * (7 - 3)
3 * (2 * (7 - 3))       (2 * 3) * (7 - 3)       2 * (3 * (7 - 3))
(2 * 3) * (7 - 3)       2 * (3 * (7 - 3))       (3 * 2) * (7 - 3)
3 * (2 * (7 - 3))
```

E. Nilai Kartu 3, 10, A, Q

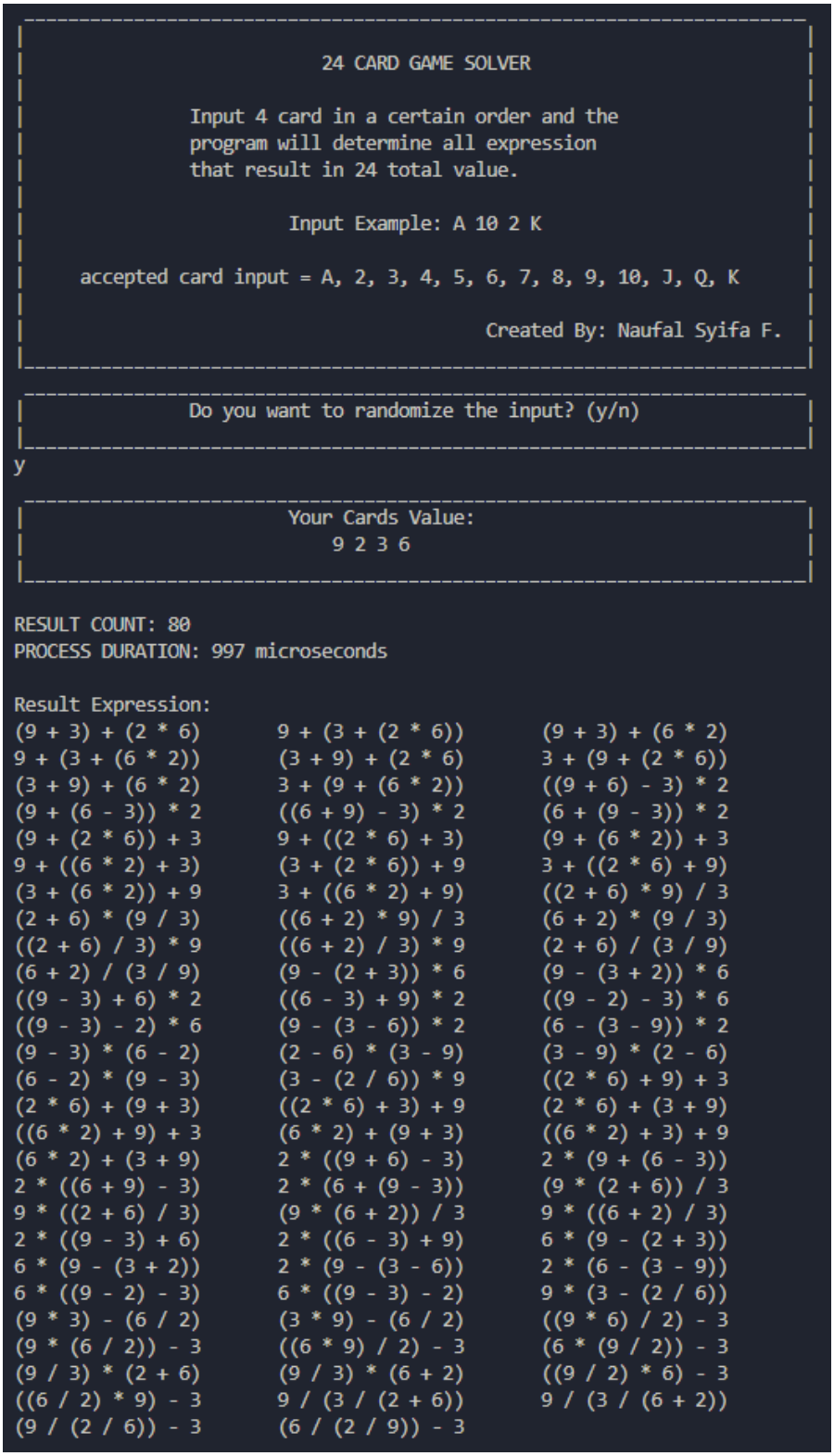


```
 ____________________________________________________________________
|                        Your Cards Value:                           |
|                            3 10 1 12                               |
|____________________________________________________________________|

RESULT COUNT: 90
PROCESS DURATION: 996 microseconds

Result Expression:
((3 + 10) + 12) - 1       (3 + (10 + 12)) - 1       (3 + 10) + (12 - 1)
3 + ((10 + 12) - 1)       3 + (10 + (12 - 1))       ((3 + 12) + 10) - 1
(3 + (12 + 10)) - 1       (3 + 12) + (10 - 1)       3 + ((12 + 10) - 1)
3 + (12 + (10 - 1))       ((10 + 3) + 12) - 1       (10 + (3 + 12)) - 1
(10 + 3) + (12 - 1)       10 + ((3 + 12) - 1)       10 + (3 + (12 - 1))
((10 + 12) + 3) - 1       (10 + (12 + 3)) - 1       (10 + 12) + (3 - 1)
10 + ((12 + 3) - 1)       10 + (12 + (3 - 1))       ((12 + 3) + 10) - 1
(12 + (3 + 10)) - 1       (12 + 3) + (10 - 1)       12 + ((3 + 10) - 1)
12 + (3 + (10 - 1))       ((12 + 10) + 3) - 1       (12 + (10 + 3)) - 1
(12 + 10) + (3 - 1)       12 + ((10 + 3) - 1)       12 + (10 + (3 - 1))
((3 + 10) - 1) + 12       (3 + (10 - 1)) + 12       3 + ((10 - 1) + 12)
((3 + 12) - 1) + 10       (3 + (12 - 1)) + 10       3 + ((12 - 1) + 10)
((10 + 3) - 1) + 12       (10 + (3 - 1)) + 12       10 + ((3 - 1) + 12)
((10 + 12) - 1) + 3       (10 + (12 - 1)) + 3       10 + ((12 - 1) + 3)
((12 + 3) - 1) + 10       (12 + (3 - 1)) + 10       12 + ((3 - 1) + 10)
((12 + 10) - 1) + 3       (12 + (10 - 1)) + 3       12 + ((10 - 1) + 3)
(3 + 10) - (1 - 12)       3 + (10 - (1 - 12))       (3 + 12) - (1 - 10)
3 + (12 - (1 - 10))       (10 + 3) - (1 - 12)       10 + (3 - (1 - 12))
(10 + 12) - (1 - 3)       10 + (12 - (1 - 3))       (12 + 3) - (1 - 10)
12 + (3 - (1 - 10))       (12 + 10) - (1 - 3)       12 + (10 - (1 - 3))
((3 - 1) + 10) + 12       (3 - 1) + (10 + 12)       ((3 - 1) + 12) + 10
(3 - 1) + (12 + 10)       ((10 - 1) + 3) + 12       (10 - 1) + (3 + 12)
((10 - 1) + 12) + 3       (10 - 1) + (12 + 3)       ((12 - 1) + 3) + 10
(12 - 1) + (3 + 10)       ((12 - 1) + 10) + 3       (12 - 1) + (10 + 3)
(3 - (1 - 10)) + 12       3 - (1 - (10 + 12))       (3 - (1 - 12)) + 10
3 - (1 - (12 + 10))       (10 - (1 - 3)) + 12       10 - (1 - (3 + 12))
(10 - (1 - 12)) + 3       10 - (1 - (12 + 3))       (12 - (1 - 3)) + 10
12 - (1 - (3 + 10))       (12 - (1 - 10)) + 3       12 - (1 - (10 + 3))
3 - ((1 - 10) - 12)       3 - ((1 - 12) - 10)       10 - ((1 - 3) - 12)
10 - ((1 - 12) - 3)       12 - ((1 - 3) - 10)       12 - ((1 - 10) - 3)
```

F. Nilai Kartu A, 7, J, 2



G. Nilai Kartu 9, 2, 3, 6



```
 ____________________________________________________________________
|                                                                    |
|                          24 CARD GAME SOLVER                       |
|                                                                    |
|              Input 4 card in a certain order and the               |
|              program will determine all expression                 |
|              that result in 24 total value.                        |
|                                                                    |
|                       Input Example: A 10 2 K                      |
|                                                                    |
|     accepted card input = A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K   |
|                                                                    |
|                                        Created By: Naufal Syifa F. |
|____________________________________________________________________|

 ____________________________________________________________________
|              Do you want to randomize the input? (y/n)             |
|____________________________________________________________________|
y
 ____________________________________________________________________
|                        Your Cards Value:                           |
|                           9 2 3 6                                  |
|____________________________________________________________________|

RESULT COUNT: 80
PROCESS DURATION: 997 microseconds

Result Expression:
(9 + 3) + (2 * 6)       9 + (3 + (2 * 6))       (9 + 3) + (6 * 2)
9 + (3 + (6 * 2))       (3 + 9) + (2 * 6)       3 + (9 + (2 * 6))
(3 + 9) + (6 * 2)       3 + (9 + (6 * 2))       ((9 + 6) - 3) * 2
(9 + (6 - 3)) * 2       ((6 + 9) - 3) * 2       (6 + (9 - 3)) * 2
(9 + (2 * 6)) + 3       9 + ((2 * 6) + 3)       (9 + (6 * 2)) + 3
9 + ((6 * 2) + 3)       (3 + (2 * 6)) + 9       3 + ((2 * 6) + 9)
(3 + (6 * 2)) + 9       3 + ((6 * 2) + 9)       ((2 + 6) * 9) / 3
(2 + 6) * (9 / 3)       ((6 + 2) * 9) / 3       (6 + 2) * (9 / 3)
((2 + 6) / 3) * 9       ((6 + 2) / 3) * 9       (2 + 6) / (3 / 9)
(6 + 2) / (3 / 9)       (9 - (2 + 3)) * 6       (9 - (3 + 2)) * 6
((9 - 3) + 6) * 2       ((6 - 3) + 9) * 2       ((9 - 2) - 3) * 6
((9 - 3) - 2) * 6       (9 - (3 - 6)) * 2       (6 - (3 - 9)) * 2
(9 - 3) * (6 - 2)       (2 - 6) * (3 - 9)       (3 - 9) * (2 - 6)
(6 - 2) * (9 - 3)       (3 - (2 / 6)) * 9       ((2 * 6) + 9) + 3
(2 * 6) + (9 + 3)       ((2 * 6) + 3) + 9       (2 * 6) + (3 + 9)
((6 * 2) + 9) + 3       (6 * 2) + (9 + 3)       ((6 * 2) + 3) + 9
(6 * 2) + (3 + 9)       2 * ((9 + 6) - 3)       2 * (9 + (6 - 3))
2 * ((6 + 9) - 3)       2 * (6 + (9 - 3))       (9 * (2 + 6)) / 3
9 * ((2 + 6) / 3)       (9 * (6 + 2)) / 3       9 * ((6 + 2) / 3)
2 * ((9 - 3) + 6)       2 * ((6 - 3) + 9)       6 * (9 - (2 + 3))
6 * (9 - (3 + 2))       2 * (9 - (3 - 6))       2 * (6 - (3 - 9))
6 * ((9 - 2) - 3)       6 * ((9 - 3) - 2)       9 * (3 - (2 / 6))
(9 * 3) - (6 / 2)       (3 * 9) - (6 / 2)       ((9 * 6) / 2) - 3
(9 * (6 / 2)) - 3       ((6 * 9) / 2) - 3       (6 * (9 / 2)) - 3
(9 / 3) * (2 + 6)       (9 / 3) * (6 + 2)       ((9 / 2) * 6) - 3
((6 / 2) * 9) - 3       9 / (3 / (2 + 6))       9 / (3 / (6 + 2))
(9 / (2 / 6)) - 3       (6 / (2 / 9)) - 3
```

# LAMPIRAN : LINK REPOSITORY

https://github.com/nomsf/Tucil1_13521050