Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Dibuat oleh: Margaretha Olivia Haryono / 13521071

I. Spesifikasi Program

Tautan Repository Github : https://github.com/margarethaolivia/Tucil1_13521071

Bahasa yang digunakan : C++

Tabel Keberjalanan Program

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	✓	
Program berhasil running	✓	
Program dapat membaca input / generate sendiri dan memberikan luaran	✓	
Solusi yang diberikan program memenuhi (berhasil mencapai 24)	✓	
Program dapat menyimpan solusi dalam file teks	✓	

II. Alur Algoritma

Pertama, program akan menampilkan menu yang tersedia untuk kartu, yaitu *input* dari *keyboard* atau *generate* secara *random*. Jika menu yang dipilih adalah *input* dari *keyboard*, maka program menerima empat masukan kartu dari *keyboard*. Jika menu yang dipilih adalah *generate* secara *random*, maka program akan memilih empat kartu secara *random*. Keempat angka masukan dari *keyboard* atau hasil *random* tersebut kemudian disimpan dalam sebuah senarai. Selanjutnya, program memanggil prosedur "permutation" yang akan melakukan permutasi secara rekursif terhadap angka-angka tersebut. Fungsi ini juga menangani kasus angka yang ganda sehingga hasil dari permutasinya dipastikan selalu unik.

Untuk setiap permutasi, dilakukan pemanggilan prosedur "eval" yang akan melakukan proses *brute force*. Pada fungsi ini, terdapat kalang *for loop* bersarang sebanyak tiga lapis yang merepresentasikan letak dari setiap operator (kalang pertama adalah operator kiri, kalang kedua adalah operator tengah, dan kalang ketiga adalah operator kanan). Setiap kalang melakukan iterasi sebanyak empat kali, yaitu sesuai dengan jumlah operasi yang valid (pertambahan, pengurangan, perkalian, dan pembagian). Pada program ini, urutan operator yang disimpan

dalam senarai operator yang valid yaitu +, -, *, dan /. Senarai ini akan digunakan untuk melakukan kombinasi operator pada setiap iterasinya. Kemudian, diinisialisasi sebuah senarai baru yang berisi urutan operasi yang akan dilakukan. Selanjutnya, dilakukan pengecekan untuk setiap pola "tanda kurung" atau urutan prioritas operasi yang dikerjakan terlebih dahulu. Terdapat lima pola kurung yang mungkin, sehingga setiap iterasi akan mengecek kelima pola tersebut.

Misal, empat kartu masukan dari *keyboard* adalah A, 2, 3, dan 4. Maka, pada iterasi pertama akan dicek lima pola berikut.

1. ((1 + 2) + 3) + 4
2. (1 + (2 + 3)) + 4
3. (1 + 2) + (3 + 4)
4. 1 + ((2 + 3) + 4)
5. 1 + (2 + (3 + 4))

Kemudian, dihitung hasil dari setiap pola tersebut. Jika dapat menghasilkan 24, maka program memanggil prosedur “save” yang bertujuan untuk menyimpan solusi tersebut ke dalam sebuah senarai yang menyimpan semua solusi yang ditemukan serta menambahkan nilai variabel yang menyimpan banyaknya solusi dengan satu. Langkah pengecekan ini diulang kembali untuk semua kombinasi operator yang valid.

Setelah menghitung semua kemungkinan kombinasi kartu, operator, serta pola, program akan menampilkan jumlah beserta semua solusi yang ditemukan. Ditampilkan juga waktu eksekusi program dalam melakukan proses *brute force*. Di akhir, program akan menanyakan apakah solusi ingin disimpan dalam sebuah *file*. Jika ya, maka pengguna dapat memasukkan nama *file* yang diinginkan dan semua solusi akan disimpan ke dalam *file* tersebut.

III. Source Program

1. Prosedur save

```
// menyimpan solusi yang ditemukan ke dalam variabel sol
void save(vector<int> cards, char ops[], int pattern, int *count, vector<string> *sol)
{
    char buffer[25]; // tempat penyimpanan string solusi sementara

    switch (pattern)
    {
    case 1:
        sprintf(buffer, "((%d %c %d) %c %d) %c %d\n", cards[0], ops[0], cards[1], ops[1], cards[2], ops[2], cards[3]);
        break;
    case 2:
        sprintf(buffer, "(%d %c (%d %c %d)) %c %d\n", cards[0], ops[0], cards[1], ops[1], cards[2], ops[2], cards[3]);
        break;
    case 3:
        sprintf(buffer, "(%d %c %d) %c (%d %c %d)\n", cards[0], ops[0], cards[1], ops[1], cards[2], ops[2], cards[3]);
        break;
    case 4:
        sprintf(buffer, "%d %c ((%d %c %d) %c %d)\n", cards[0], ops[0], cards[1], ops[1], cards[2], ops[2], cards[3]);
        break;
    case 5:
        sprintf(buffer, "%d %c (%d %c (%d %c %d))\n", cards[0], ops[0], cards[1], ops[1], cards[2], ops[2], cards[3]);
        break;
    default:
        cerr << "Pola tidak valid!" << endl;
        throw -1;
    }

    *count += 1;            // banyaknya solusi ditambah 1
    sol->push_back(buffer); // menyimpan string solusi
}
```

2. Fungsi calculate

```
// mengembalikan hasil operasi sesuai operator
float calculate(char op, float num1, float num2)
{
    switch (op)
    {
    case '+':
        return num1 + num2;
    case '-':
        return num1 - num2;
    case '*':
        return num1 * num2;
    case '/':
        return num1 / num2;
    default:
        cerr << "Operator tidak valid!" << endl;
        throw -1;
    }
}
```

3. Prosedur eval

```
// melakukan brute force terhadap operator yang valid dan kelima pola
void eval(char opvalid[], vector<int> cards, float GOAL, int *count, vector<string> *sol)
{
    int i, j, k;
    float res;
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < 4; j++)
        {
            for (k = 0; k < 4; k++)
            {
                char ops[3] = {opvalid[i], opvalid[j], opvalid[k]};

                // pattern 1
                res = calculate(ops[0], cards[0], cards[1]);
                res = calculate(ops[1], res, cards[2]);
                res = calculate(ops[2], res, cards[3]);
                if (res == GOAL)
                {
                    save(cards, ops, 1, count, sol);
                }

                // pattern 2
                res = calculate(ops[1], cards[1], cards[2]);
                res = calculate(ops[0], cards[0], res);
                res = calculate(ops[2], res, cards[3]);
                if (res == GOAL)
                {
                    save(cards, ops, 2, count, sol);
                }
```

```
                // pattern 3
                float res1 = calculate(ops[0], cards[0], cards[1]);
                float res2 = calculate(ops[2], cards[2], cards[3]);
                res = calculate(ops[1], res1, res2);
                if (res == GOAL)
                {
                    save(cards, ops, 3, count, sol);
                }

                // pattern 4
                res = calculate(ops[1], cards[1], cards[2]);
                res = calculate(ops[2], res, cards[3]);
                res = calculate(ops[0], cards[0], res);
                if (res == GOAL)
                {
                    save(cards, ops, 4, count, sol);
                }

                // pattern 5
                res = calculate(ops[2], cards[2], cards[3]);
                res = calculate(ops[1], cards[1], res);
                res = calculate(ops[0], cards[0], res);
                if (res == GOAL)
                {
                    save(cards, ops, 5, count, sol);
                }
            }
        }
    }
}
```

4. Prosedur swap

```
// menukar variabel
void swap(float &a, float &b)
{
    float temp = a;
    a = b;
    b = temp;
}
```

5. Fungsi isSwap

```
// mengecek apakah suatu angka perlu ditukar atau tidak (menangani kasus jika ada masukan angka yang ganda)
// jika angka sama (ganda), maka mengembalikan 0 (false) -> tidak perlu ditukar
// jika tidak, maka mengembalikan 1 (true)
bool isSwap(vector<int> nums, int start, int curr)
{
    for (int i = start; i < curr; i++)
    {
        if (nums[i] == nums[curr])
        {
            return 0;
        }
    }
    return 1;
}
```

6. Prosedur permutation

```
// melakukan permutasi terhadap angka masukan, serta memanggil fungsi eval (untuk brute force) pada setiap permutasi yang ada
void permutation(vector<int> nums, int idx, int n, char opvalid[], int *count, vector<string> *sol)
{
    if (idx >= n)
    {
        eval(opvalid, nums, 24, count, sol);
        return;
    }

    for (int i = idx; i < n; i++)
    {
        bool check = isSwap(nums, idx, i);
        if (check)
        {
            swap(nums[idx], nums[i]);
            permutation(nums, idx + 1, n, opvalid, count, sol);
            swap(nums[idx], nums[i]);
        }
    }
}
```

7. Main Program

```
int main()
{
    const float GOAL = 24.0; // hasil yang diinginkan

    string menu, input, save, filename, *result; // variabel untuk input

    int count = 0; // banyaknya solusi

    vector<string> sol; // vector penyimpanan solusi
    vector<int> cards;  // vector penyimpanan angka/kartu masukan

    string cardvalid[13] = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"}; // array kartu yang valid
    char opvalid[4] = {'+', '-', '*', '/'};                                                    // array operator yang valid

    int i; // variabel untuk iterasi
```

```
cout << "SELAMAT DATANG DI \"24 CARD GAME\" SOLVER" << endl
     << endl;

cout << "Pilihan:" << endl;
cout << "1. Input dari keyboard" << endl;
cout << "2. Generate dari program" << endl
     << endl;

// input dan validasi menu
do
{
    cout << "Pilih menu (1/2) : ";
    cin >> menu;
} while (menu != "1" && menu != "2" && printf("Masukan tidak sesuai!\n"));

cout << endl;
```

```
if (menu == "1") // input dari keyboard
{
    for (i = 0; i < 4; i++)
    {
        // input dan validasi kartu
        do
        {
            cout << "Masukkan kartu ke-" << i + 1 << ": ";
            cin >> input;
            result = find(begin(cardvalid), end(cardvalid), input);
        } while (result == end(cardvalid) && printf("Masukan tidak sesuai!\n"));

        cards.push_back(distance(cardvalid, result) + 1);
    }
    cout << endl;
}
else // menu == 2 -> random
{
    // menggunakan current time untuk seed random
    srand(time(0));

    // angka yang valid (1-13)
    int N = 13;

    for (i = 0; i < 4; i++)
    {
        cards.push_back((rand() % N) + 1);
    }
}
```

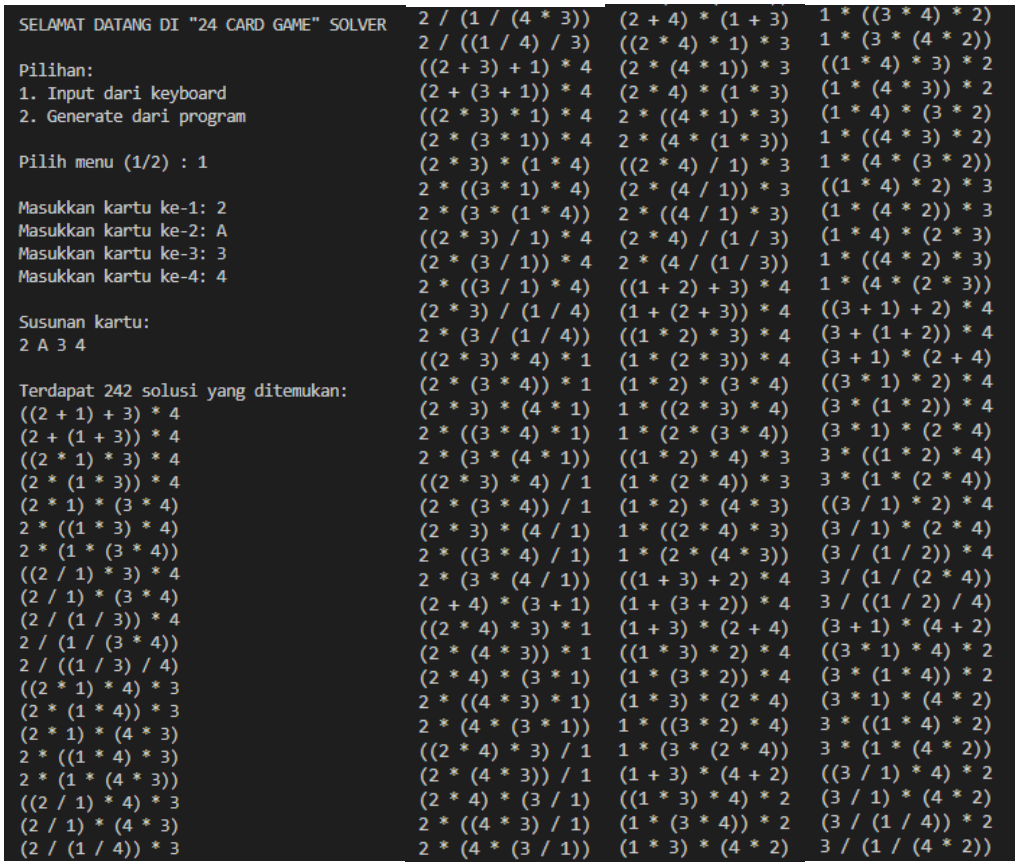
```
// output susunan kartu masukan
cout << "Susunan kartu:" << endl;
for (auto x : cards)
{
    cout << cardvalid[x - 1] << " ";
}

cout << endl
     << endl;

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

// proses bruteforce
permutation(cards, 0, 4, opvalid, &count, &sol);

// end stopwatch
auto end = chrono::high_resolution_clock::now();
```

```
    // output solusi
    if (count == 0)
    {
        sol.push_back("Tidak ada solusi yang ditemukan\n");
        cout << sol[0];
    }
    else
    {
        cout << "Terdapat " << count << " solusi yang ditemukan:" << endl;

        for (auto x : sol)
        {
            cout << x;
        }
    }

    // output waktu eksekusi program
    auto time = chrono::duration_cast<chrono::microseconds>(end - start);
    cout << endl
         << "Waktu eksekusi: " << time.count() << " mikrosekon" << endl
         << endl;
```

```
    // menyimpan solusi ke file txt
    do
    {
        cout << "Apakah ingin menyimpan solusi? (y/n) : ";
        cin >> save;
    } while (save != "y" && save != "n" && printf("Masukan tidak sesuai!\n"));

    if (save == "y")
    {
        cout << endl
             << "Masukkan nama file yang diinginkan : ";
        cin >> filename;
        fstream file;
        file.open("test/" + filename + ".txt", ios_base::out);

        for (i = 0; i < sol.size(); i++)
        {
            file << sol[i];
        }

        cout << "Berhasil menyimpan solusi pada test/" << filename << ".txt" << endl;
    }

    cout << endl
         << "Terima kasih telah menggunakan \"24 Card Game\" Solver" << endl;

    return 0;
}
```

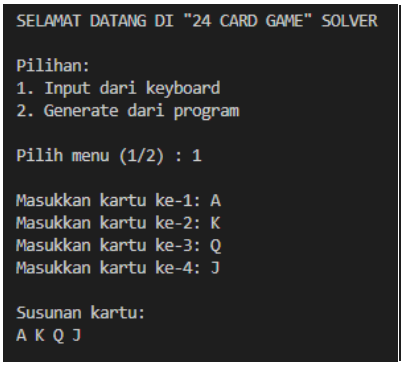
IV. *Screenshot* Uji Kasus

1. Input Keyboard, Solusi Banyak (2 A 3 4)

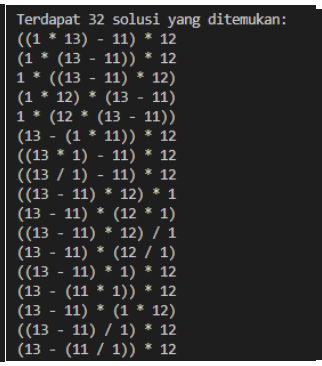


```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 1

Masukkan kartu ke-1: 2
Masukkan kartu ke-2: A
Masukkan kartu ke-3: 3
Masukkan kartu ke-4: 4

Susunan kartu:
2 A 3 4

Terdapat 242 solusi yang ditemukan:
((2 + 1) + 3) * 4
(2 + (1 + 3)) * 4
((2 * 1) * 3) * 4
(2 * (1 * 3)) * 4
(2 * 1) * (3 * 4)
2 * ((1 * 3) * 4)
2 * (1 * (3 * 4))
((2 / 1) * 3) * 4
(2 / 1) * (3 * 4)
(2 / (1 / 3)) * 4
2 / (1 / (3 * 4))
2 / ((1 / 3) / 4)
((2 * 1) * 4) * 3
(2 * (1 * 4)) * 3
(2 * 1) * (4 * 3)
2 * ((1 * 4) * 3)
2 * (1 * (4 * 3))
((2 / 1) * 4) * 3
(2 / 1) * (4 * 3)
(2 / (1 / 4)) * 3
2 / (1 / (4 * 3))
2 / ((1 / 4) / 3)
((2 + 3) + 1) * 4
(2 + (3 + 1)) * 4
((2 * 3) * 1) * 4
(2 * (3 * 1)) * 4
(2 * 3) * (1 * 4)
2 * ((3 * 1) * 4)
2 * (3 * (1 * 4))
((2 * 3) / 1) * 4
(2 * (3 / 1)) * 4
2 * ((3 / 1) * 4)
(2 * 3) / (1 / 4)
2 * (3 / (1 / 4))
((2 * 3) * 4) * 1
(2 * (3 * 4)) * 1
(2 * 3) * (4 * 1)
2 * ((3 * 4) * 1)
2 * (3 * (4 * 1))
((2 * 3) * 4) / 1
(2 * (3 * 4)) / 1
(2 * 3) * (4 / 1)
2 * ((3 * 4) / 1)
2 * (3 * (4 / 1))
(2 + 4) * (3 + 1)
((2 * 4) * 3) * 1
(2 * (4 * 3)) * 1
(2 * 4) * (3 * 1)
2 * ((4 * 3) * 1)
2 * (4 * (3 * 1))
((2 * 4) * 3) / 1
(2 * (4 * 3)) / 1
(2 * 4) * (3 / 1)
2 * ((4 * 3) / 1)
2 * (4 * (3 / 1))
(2 + 4) * (1 + 3)
((2 * 4) * 1) * 3
(2 * (4 * 1)) * 3
(2 * 4) * (1 * 3)
2 * ((4 * 1) * 3)
2 * (4 * (1 * 3))
((2 / 1) * 4) * 3
(2 / 1) * (4 * 3)
(2 / (1 / 4)) * 3
2 / (1 / (4 * 3))
2 / ((1 / 4) / 3)
1 * ((3 * 4) * 2)
1 * (3 * (4 * 2))
((1 * 4) * 3) * 2
(1 * (4 * 3)) * 2
(1 * 4) * (3 * 2)
1 * ((4 * 3) * 2)
1 * (4 * (3 * 2))
((1 * 4) * 2) * 3
(1 * (4 * 2)) * 3
(1 * 4) * (2 * 3)
1 * ((4 * 2) * 3)
1 * (4 * (2 * 3))
((3 + 1) + 2) * 4
(3 + (1 + 2)) * 4
(3 + 1) * (2 + 4)
((3 * 1) * 2) * 4
(3 * (1 * 2)) * 4
(3 * 1) * (2 * 4)
3 * ((1 * 2) * 4)
3 * (1 * (2 * 4))
((3 / 1) * 2) * 4
(3 / 1) * (2 * 4)
(3 / (1 / 2)) * 4
3 / (1 / (2 * 4))
3 / ((1 / 2) / 4)
(3 + 1) * (4 + 2)
((3 * 1) * 4) * 2
(3 * (1 * 4)) * 2
(3 * 1) * (4 * 2)
3 * ((1 * 4) * 2)
3 * (1 * (4 * 2))
3 / ((1 / 4) / 2)
((3 + 2) + 1) * 4
(3 + (2 + 1)) * 4
((3 * 2) * 1) * 4
(3 * (2 * 1)) * 4
(3 * 2) * (1 * 4)
3 * ((2 * 1) * 4)
3 * (2 * (1 * 4))
((3 * 2) / 1) * 4
(3 * (2 / 1)) * 4
3 * ((2 / 1) * 4)
(3 * 2) / (1 / 4)
3 * (2 / (1 / 4))
((3 * 2) * 4) * 1
(3 * (2 * 4)) * 1
(3 * 2) * (4 * 1)
3 * ((2 * 4) * 1)
3 * (2 * (4 * 1))
((3 * 2) * 4) / 1
(3 * (2 * 4)) / 1
(3 * 2) * (4 / 1)
3 * ((2 * 4) / 1)
3 * (2 * (4 / 1))
((3 * 4) * 2) * 1
(3 * (4 * 2)) * 1
(3 * 4) * (2 * 1)
3 * ((4 * 2) * 1)
3 * (4 * (2 * 1))
((3 * 4) * 2) / 1
(3 * (4 * 2)) / 1
(3 * 4) * (2 / 1)
3 * ((4 * 2) / 1)
3 * (4 * (2 / 1))
((3 * 4) * 1) * 2
(3 * (4 * 1)) * 2
(3 * 4) * (1 * 2)
3 * ((4 * 1) * 2)
3 * (4 * (1 * 2))
((3 * 4) / 1) * 2
(3 * (4 / 1)) * 2
3 * ((4 / 1) * 2)
(3 * 4) / (1 / 2)
3 * (4 / (1 / 2))
4 * ((1 + 3) + 2)
4 * (1 + (3 + 2))
((4 * 1) * 3) * 2
(4 * (1 * 3)) * 2
(4 * 1) * (3 * 2)
4 * ((1 * 3) * 2)
4 * (1 * (3 * 2))
((4 / 1) * 3) * 2
(4 / 1) * (3 * 2)
(4 / (1 / 3)) * 2
4 / (1 / (3 * 2))
4 / ((1 / 3) / 2)
4 * ((1 + 2) + 3)
4 * (1 + (2 + 3))
((4 * 1) * 2) * 3
(4 * (1 * 2)) * 3
(4 * 1) * (2 * 3)
4 * ((1 * 2) * 3)
4 * (1 * (2 * 3))
((4 / 1) * 2) * 3
(4 / 1) * (2 * 3)
(4 / (1 / 2)) * 3
4 / (1 / (2 * 3))
4 / ((1 / 2) / 3)
4 * ((3 + 1) + 2)
4 * (3 + (1 + 2))
((4 * 3) * 1) * 2
(4 * (3 * 1)) * 2
(4 * 3) * (1 * 2)
4 * ((3 * 1) * 2)
4 * (3 * (1 * 2))
((4 * 3) / 1) * 2
(4 * (3 / 1)) * 2
4 * ((3 / 1) * 2)
(4 * 3) / (1 / 2)
4 * (3 / (1 / 2))
4 * ((3 + 2) + 1)
4 * (3 + (2 + 1))
((4 * 3) * 2) * 1
(4 * (3 * 2)) * 1
(4 * 3) * (2 * 1)
4 * ((3 * 2) * 1)
4 * (3 * (2 * 1))
((4 * 3) * 2) / 1
(4 * (3 * 2)) / 1
(4 * 3) * (2 / 1)
4 * ((3 * 2) / 1)
4 * (3 * (2 / 1))
(4 + 2) * (3 + 1)
4 * ((2 + 3) + 1)
4 * (2 + (3 + 1))
((4 * 2) * 3) * 1
(4 * (2 * 3)) * 1
(4 * 2) * (3 * 1)
4 * ((2 * 3) * 1)
4 * (2 * (3 * 1))
((4 * 2) * 3) / 1
(4 * (2 * 3)) / 1
(4 * 2) * (3 / 1)
4 * ((2 * 3) / 1)
4 * (2 * (3 / 1))
(4 + 2) * (1 + 3)
4 * ((2 + 1) + 3)
4 * (2 + (1 + 3))
((4 * 2) * 1) * 3
(4 * (2 * 1)) * 3
(4 * 2) * (1 * 3)
4 * ((2 * 1) * 3)
4 * (2 * (1 * 3))
((4 * 2) / 1) * 3
(4 * (2 / 1)) * 3
4 * ((2 / 1) * 3)
(4 * 2) / (1 / 3)
4 * (2 / (1 / 3))

Waktu eksekusi: 2001 mikrosekon

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

Masukkan nama file yang diinginkan : tc1
Berhasil menyimpan solusi pada test/tc1.txt

Terima kasih telah menggunakan "24 Card Game" Solver
```

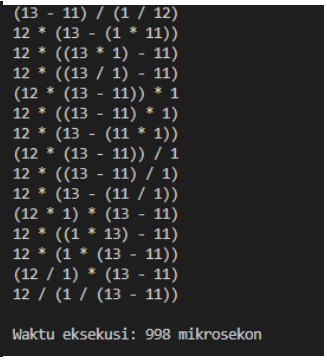
2. Input Keyboard, Solusi Sedang (A K Q J)



```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 1

Masukkan kartu ke-1: A
Masukkan kartu ke-2: K
Masukkan kartu ke-3: Q
Masukkan kartu ke-4: J

Susunan kartu:
A K Q J
```

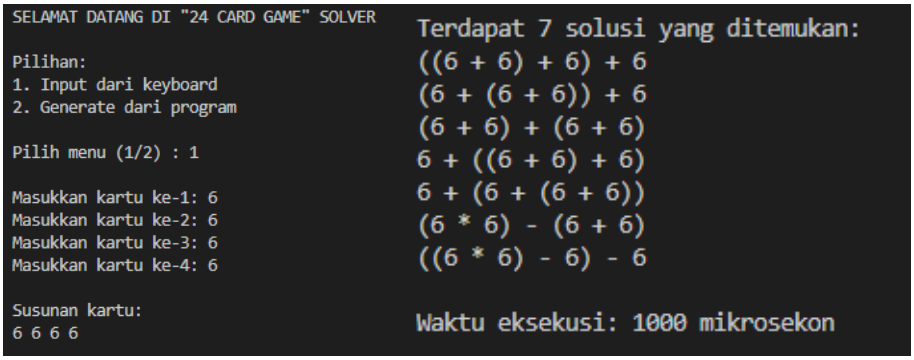


```
Terdapat 32 solusi yang ditemukan:
((1 * 13) - 11) * 12
(1 * (13 - 11)) * 12
1 * ((13 - 11) * 12)
(1 * 12) * (13 - 11)
1 * (12 * (13 - 11))
(13 - (1 * 11)) * 12
((13 * 1) - 11) * 12
((13 / 1) - 11) * 12
((13 - 11) * 12) * 1
(13 - 11) * (12 * 1)
((13 - 11) * 12) / 1
(13 - 11) * (12 / 1)
((13 - 11) * 1) * 12
(13 - (11 * 1)) * 12
(13 - 11) * (1 * 12)
((13 - 11) / 1) * 12
(13 - (11 / 1)) * 12
```



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

Waktu eksekusi: 998 mikrosekon
```

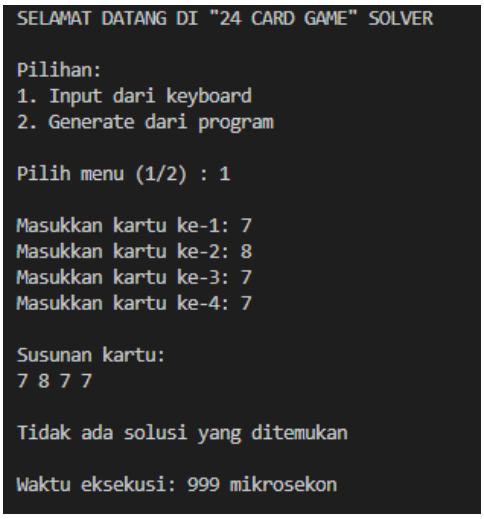
3. Input Keyboard, Solusi Sedikit (6 6 6 6)



```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 1

Masukkan kartu ke-1: 6
Masukkan kartu ke-2: 6
Masukkan kartu ke-3: 6
Masukkan kartu ke-4: 6

Susunan kartu:
6 6 6 6
```

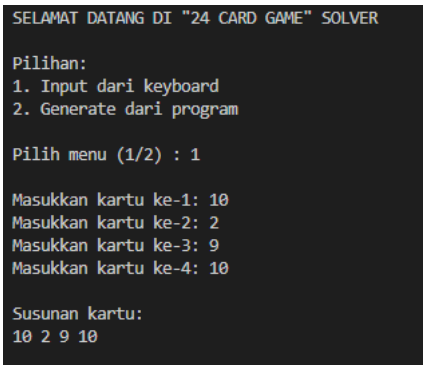
```
Terdapat 7 solusi yang ditemukan:
((6 + 6) + 6) + 6
(6 + (6 + 6)) + 6
(6 + 6) + (6 + 6)
6 + ((6 + 6) + 6)
6 + (6 + (6 + 6))
(6 * 6) - (6 + 6)
((6 * 6) - 6) - 6

Waktu eksekusi: 1000 mikrosekon
```

4. Input Keyboard, Tidak Ada Solusi (7 8 7 7)

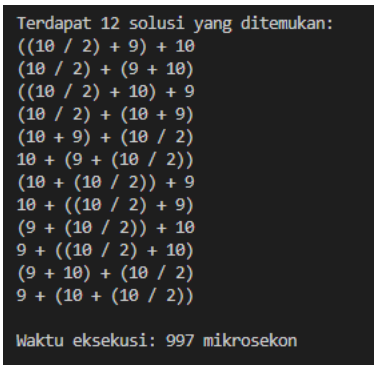


```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 1

Masukkan kartu ke-1: 7
Masukkan kartu ke-2: 8
Masukkan kartu ke-3: 7
Masukkan kartu ke-4: 7

Susunan kartu:
7 8 7 7

Tidak ada solusi yang ditemukan

Waktu eksekusi: 999 mikrosekon
```

5. Input Keyboard, Dua Kartu Kembar (10 2 9 10)

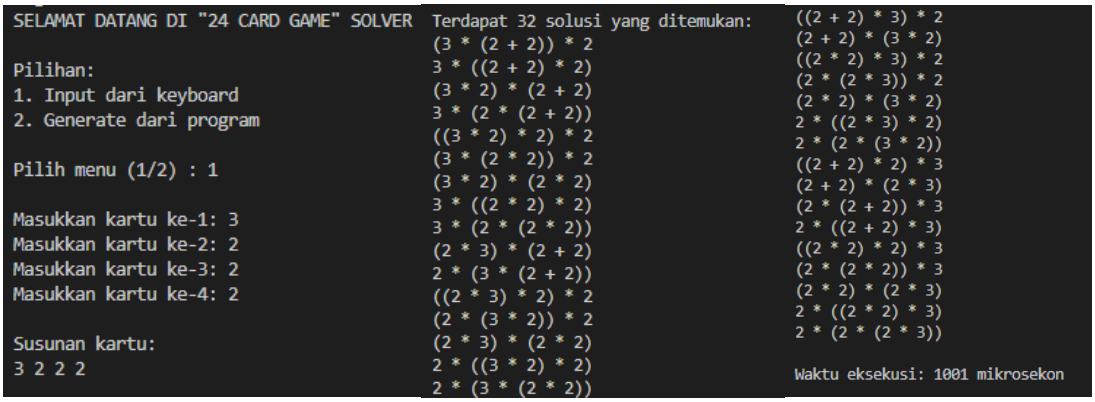


```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 1

Masukkan kartu ke-1: 10
Masukkan kartu ke-2: 2
Masukkan kartu ke-3: 9
Masukkan kartu ke-4: 10

Susunan kartu:
10 2 9 10
```



```
Terdapat 12 solusi yang ditemukan:
((10 / 2) + 9) + 10
(10 / 2) + (9 + 10)
((10 / 2) + 10) + 9
(10 / 2) + (10 + 9)
(10 + 9) + (10 / 2)
10 + (9 + (10 / 2))
(10 + (10 / 2)) + 9
10 + ((10 / 2) + 9)
(9 + (10 / 2)) + 10
9 + ((10 / 2) + 10)
(9 + 10) + (10 / 2)
9 + (10 + (10 / 2))

Waktu eksekusi: 997 mikrosekon
```

6. Input Keyboard, Tiga Kartu Kembar (3 2 2 2)

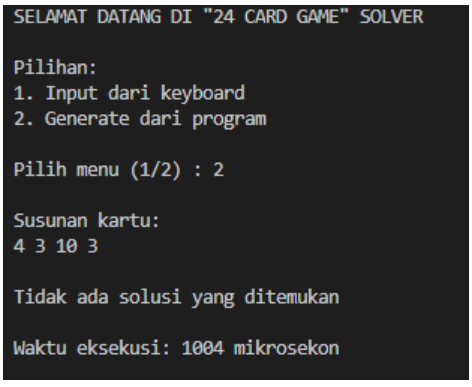


```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 1

Masukkan kartu ke-1: 3
Masukkan kartu ke-2: 2
Masukkan kartu ke-3: 2
Masukkan kartu ke-4: 2

Susunan kartu:
3 2 2 2
```

```
Terdapat 32 solusi yang ditemukan:
(3 * (2 + 2)) * 2
3 * ((2 + 2) * 2)
(3 * 2) * (2 + 2)
3 * (2 * (2 + 2))
((3 * 2) * 2) * 2
(3 * (2 * 2)) * 2
(3 * 2) * (2 * 2)
3 * ((2 * 2) * 2)
3 * (2 * (2 * 2))
(2 * 3) * (2 + 2)
2 * (3 * (2 + 2))
((2 * 3) * 2) * 2
(2 * (3 * 2)) * 2
(2 * 3) * (2 * 2)
2 * ((3 * 2) * 2)
2 * (3 * (2 * 2))
```

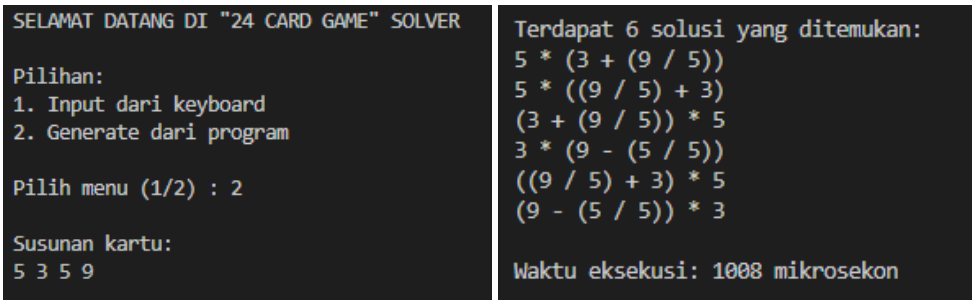
```
((2 + 2) * 3) * 2
(2 + 2) * (3 * 2)
((2 * 2) * 3) * 2
(2 * (2 * 3)) * 2
(2 * 2) * (3 * 2)
2 * ((2 * 3) * 2)
2 * (2 * (3 * 2))
((2 + 2) * 2) * 3
(2 + 2) * (2 * 3)
(2 * (2 + 2)) * 3
2 * ((2 + 2) * 3)
((2 * 2) * 2) * 3
(2 * (2 * 2)) * 3
(2 * 2) * (2 * 3)
2 * ((2 * 2) * 3)
2 * (2 * (2 * 3))

Waktu eksekusi: 1001 mikrosekon
```

7. Menu Random, Tidak Ada Solusi



```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 2

Susunan kartu:
4 3 10 3

Tidak ada solusi yang ditemukan

Waktu eksekusi: 1004 mikrosekon
```

8. Menu Random, Ada Solusi



```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 2

Susunan kartu:
5 3 5 9
```

```
Terdapat 6 solusi yang ditemukan:
5 * (3 + (9 / 5))
5 * ((9 / 5) + 3)
(3 + (9 / 5)) * 5
3 * (9 - (5 / 5))
((9 / 5) + 3) * 5
(9 - (5 / 5)) * 3

Waktu eksekusi: 1008 mikrosekon
```

9. Menyimpan Solusi

```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 2

Susunan kartu:
9 2 7 5
```

```
Terdapat 8 solusi yang ditemukan:
(7 * 5) - (9 + 2)
((7 * 5) - 9) - 2
(7 * 5) - (2 + 9)
((7 * 5) - 2) - 9
(5 * 7) - (2 + 9)
((5 * 7) - 2) - 9
(5 * 7) - (9 + 2)
((5 * 7) - 9) - 2

Waktu eksekusi: 1002 mikrosekon
```

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

Masukkan nama file yang diinginkan : tc8
Berhasil menyimpan solusi pada test/tc8.txt

Terima kasih telah menggunakan "24 Card Game" Solver
```

10. Validasi Masukan
    a. Menu

```
SELAMAT DATANG DI "24 CARD GAME" SOLVER

Pilihan:
1. Input dari keyboard
2. Generate dari program

Pilih menu (1/2) : 0
Masukan tidak sesuai!
Pilih menu (1/2) : menu
Masukan tidak sesuai!
Pilih menu (1/2) : ?
Masukan tidak sesuai!
Pilih menu (1/2) : 1

Masukkan kartu ke-1:
```

    b. Kartu

```
Pilih menu (1/2) : 1

Masukkan kartu ke-1: a
Masukan tidak sesuai!
Masukkan kartu ke-1: A
Masukkan kartu ke-2: 12
Masukan tidak sesuai!
Masukkan kartu ke-2: Q
Masukkan kartu ke-3: ++
Masukan tidak sesuai!
Masukkan kartu ke-3: 2
Masukkan kartu ke-4: 2

Susunan kartu:
A Q 2 2
```

c. Menyimpan Solusi

```
Apakah ingin menyimpan solusi? (y/n) : tidak
Masukan tidak sesuai!
Apakah ingin menyimpan solusi? (y/n) : =
Masukan tidak sesuai!
Apakah ingin menyimpan solusi? (y/n) : 1
Masukan tidak sesuai!
Apakah ingin menyimpan solusi? (y/n) : n

Terima kasih telah menggunakan "24 Card Game" Solver
```