Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*

Oleh:

Bagas Aryo Seto

K01 / 13521081

PROGRAM STUDIO TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG 2023

A. Algoritma *Brute Force*

Algoritma *brute force* merupakan pendekatan yang lempang (*straightforward*) dalam memecahkan suatu persoalan. Algoritma *brute force* biasanya didasarkan pada definisi atau konsep yang dilibatkan serta pernyataan pada persoalan itu sendiri. Algoritma ini menyelesaikan masalah dengan meninjau semua kasus yang meuncul pada persoalan terkait.

Dalam menyelesaikan permainan kartu 24, dapat digunakan algoritma *brute force*. Berikut langkah-langkah algoritma *brute force* yang dibuat:

1. Tentukan semua permutasi angka kartu yang mungkin. Terdapat total 24 kemungkinan permutasi yang berbeda untuk empat kartu yang berbeda.
2. Tentukan semua persamaan yang mungkin dibuat dengan setiap permutasi kartu yang ada dengan operator yang valid. Persamaan yang dibuat memiliki pengelompokan sebagai berikut:
   a. ((X o X) o X) o X
   b. (X o (X o X)) o X
   c. X o ((X o X) o X)
   d. X o (X o (X o X))
   e. (X o X) o (X o X)

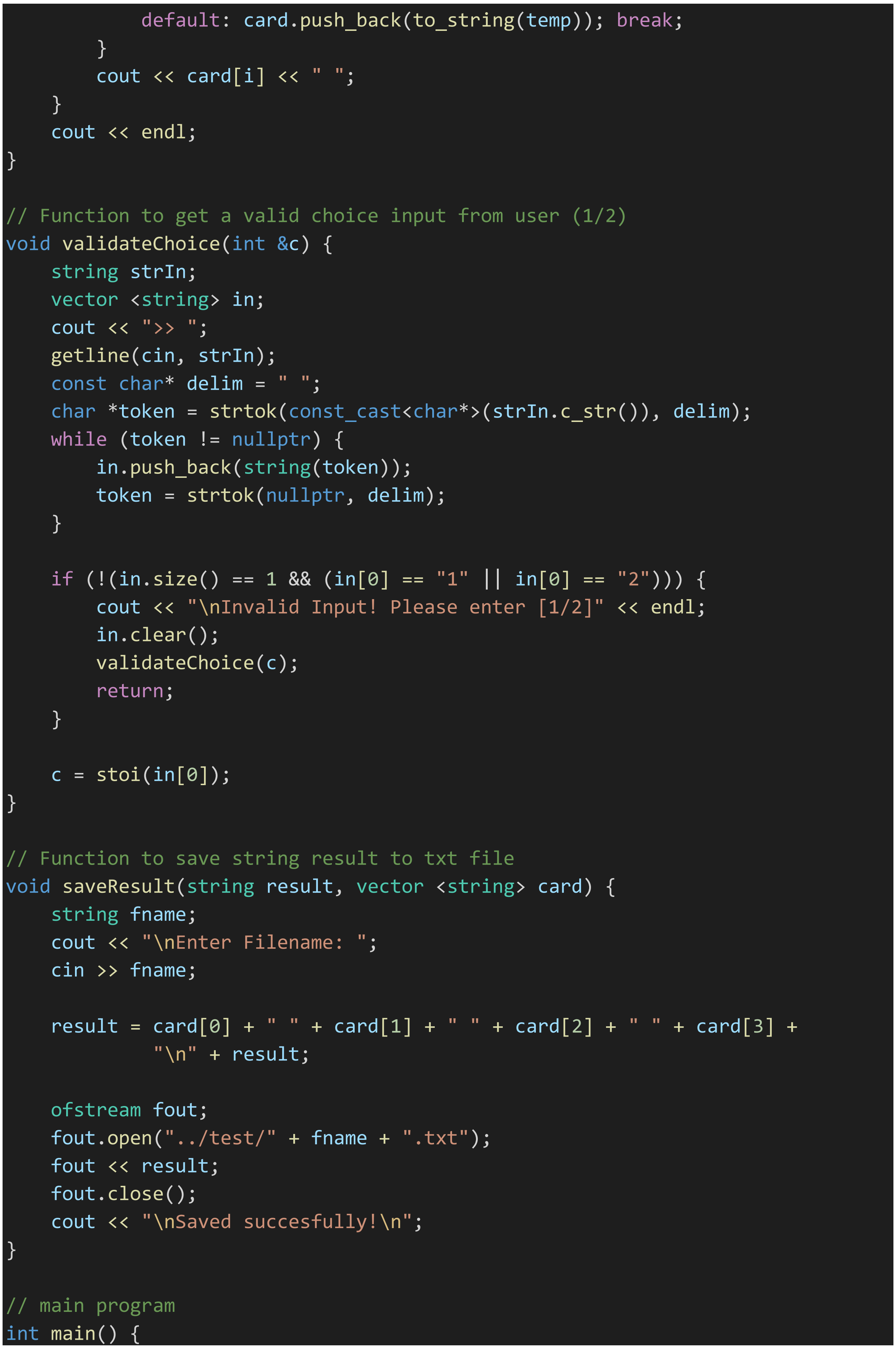
   Dengan ‘X’ merupakan nilai dari kartu dan ‘o’ merupakan operator. Operator yang dapat digunakan meliputi operator penjumlahan (+), pengurangan (-), perkalian (*), dan pembagian (/).

3. Hitung nilai semua persamaan, jika persamaan bernilai 24 maka persamaan tersebut merupakan solusi dari permainan 24.
4. Ulangi langkah dua dan tiga hingga semua solusi didapatkan.
5. Hapus solusi yang merupakan duplikat apabila terdapat kartu yang sama.

B. Source Program

```cpp
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
using namespace std::chrono;

// Function to calculate arithmetic operations
double calc(double a, char op, double b){
    switch(op){
        case '+': return a + b;
        case '-': return a - b;
        case '*': return a * b;
        case '/': return a / b;
    }
}

// Function to solve all possible combination for 24 game
void solve(vector <string> sCard, string &result) {

    // Process the string vector to int vector of cards
    vector <int> iCard;
    for (int i = 0; i < 4; i++) {
        if (sCard[i] == "A") {
            iCard.push_back(1);
            sCard[i] = "1";
        }
        else if (sCard[i] == "J") {
            iCard.push_back(11);
            sCard[i] = "11";
        }
        else if (sCard[i] == "Q") {
            iCard.push_back(12);
            sCard[i] = "12";
        }
        else if (sCard[i] == "K") {
            iCard.push_back(13);
            sCard[i] = "13";
        }
        else {
            iCard.push_back(stoi(sCard[i]));
        }
    }

    // get start time for brute force algorithm
    auto start = high_resolution_clock::now();

    // all possible position/permutation for the cards
    int pos[24][4] = {
        {0,1,2,3}, {0,1,3,2}, {0,2,1,3}, {0,2,3,1}, {0,3,1,2}, {0,3,2,1},
        {1,0,2,3}, {1,0,3,2}, {1,2,0,3}, {1,2,3,0}, {1,3,0,2}, {1,3,2,0},
        {2,0,1,3}, {2,0,3,1}, {2,1,0,3}, {2,1,3,0}, {2,3,0,1}, {2,3,1,0},
        {3,0,1,2}, {3,0,2,1}, {3,1,0,2}, {3,1,2,0}, {3,2,0,1}, {3,2,1,0}};

    string sym = "/*-+";
    string ops[64], expr;
    vector <string> vecRes;
    double val;

    // brute force all possible cards and symbol combinations
    for (int x = 0; x < 24; x++) {
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                for (int k = 0; k < 4; k++) {

                    // brute force all possible grouping for the equations
                    double temp;
                    temp = calc(calc(calc(iCard[pos[x][0]], sym[i],
                           iCard[pos[x][1]]), sym[j], iCard[pos[x][2]]),
                           sym[k], iCard[pos[x][3]]);
                    if (temp == 24) {
                        vecRes.push_back("((" + sCard[pos[x][0]] + " " +
                               sym[i] + " " + sCard[pos[x][1]] + ") " +
                               sym[j] + " " + sCard[pos[x][2]] + ") " + sym[k]
                               + " " + sCard[pos[x][3]]);
                    }

                    temp = calc(calc(iCard[pos[x][0]], sym[i],
                           calc(iCard[pos[x][1]], sym[j], iCard[pos[x][2]])),
                           sym[k], iCard[pos[x][3]]);
                    if (temp == 24) {
                        vecRes.push_back("(" + sCard[pos[x][0]] + " " + sym[i]
                               + " (" + sCard[pos[x][1]] + " " + sym[j] + " "
                               + sCard[pos[x][2]] + ")) " + sym[k] + " " +
                               sCard[pos[x][3]]);
                    }

                    temp = calc(iCard[pos[x][0]], sym[i],
                           calc(calc(iCard[pos[x][1]], sym[j],
                           iCard[pos[x][2]]), sym[k], iCard[pos[x][3]]));
                    if (temp == 24) {
                        vecRes.push_back(sCard[pos[x][0]] + " " + sym[i] +
                               " ((" + sCard[pos[x][1]] + " " + sym[j] +
                               " " + sCard[pos[x][2]] + ") " + sym[k] + " " +
                               sCard[pos[x][3]] + ")");
                    }
                    temp = calc(iCard[pos[x][0]], sym[i],
                           calc(iCard[pos[x][1]], sym[j],
                           calc(iCard[pos[x][2]], sym[k], iCard[pos[x][3]])));
                    if (temp == 24) {
                        vecRes.push_back(sCard[pos[x][0]] + " " + sym[i] +
                               " (" + sCard[pos[x][1]] + " " + sym[j] +
                               " (" + sCard[pos[x][2]] + " " + sym[k] + " " +
                               sCard[pos[x][3]] + "))");
                    }

                    temp = calc(calc(iCard[pos[x][0]], sym[i],
                           iCard[pos[x][1]]), sym[j],
                           calc(iCard[pos[x][2]], sym[k], iCard[pos[x][3]]));
                    if (temp == 24) {
                        vecRes.push_back("(" + sCard[pos[x][0]] + " " + sym[i]
                               + " " + sCard[pos[x][1]] + ") " + sym[j] +
                               " (" + sCard[pos[x][2]] + " " + sym[k] + " " +
                               sCard[pos[x][3]] + ")");
                    }
                }
            }
        }
    }

    // delete duplicated result
    unordered_set <string> undupe(vecRes.begin(), vecRes.end());
    vecRes.assign(undupe.begin(), undupe.end());

    // get stop time and execution time for the algorithm
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<milliseconds>(stop - start);

    // result processing
    result += to_string(vecRes.size()) + " solution(s) found\n";
    for (auto i = vecRes.begin(); i != vecRes.end(); ++i) result += *i + "\n";
    result += "time taken: " + to_string(duration.count()) + " ms\n";
}

// Function to get cards from user input
void cardInput(vector <string> &card) {
    // get user input
    cout << "\nPlease enter 4 valid values [A,2,3,..,Q,K]
              separated with space\n>> ";
    string strIn;
    getline(cin, strIn);
    const char* delim = " ";
    char *token = strtok(const_cast<char*>(strIn.c_str()), delim);
```

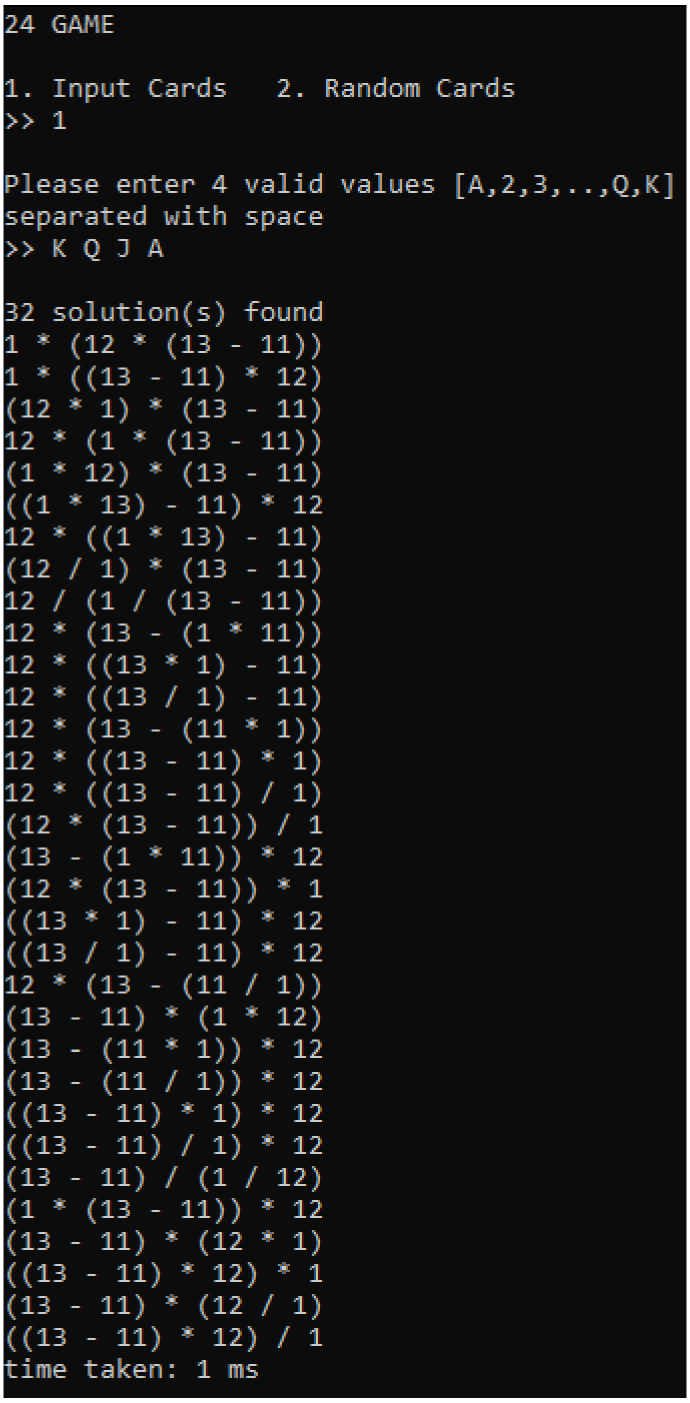
```cpp
    while (token != nullptr) {
        card.push_back(string(token));
        token = strtok(nullptr, delim);
    }

    // input validation
    if (card.size() != 4) {
        cout << "Invalid Input!\n";
        card.clear();
        cardInput(card);
        return;
    }

    for (int i = 0; i < 4; i++) {
        try{
            int x = stoi(card[i]);
            if (x < 2 || x > 10) {
                cout << "Invalid Input!\n";
                card.clear();
                cardInput(card);
                return;
            }
        }
        catch(const exception& e) {
            if (!(card[i] == "A" || card[i] == "J" ||
                card[i] == "Q" || card[i] == "K")) {
                cout << "Invalid Input!\n";
                card.clear();
                cardInput(card);
                return;
            }
        }
    }
}

// Function to generate random cards
void cardRandom(vector <string> &card) {
    srand(time(0));

    cout << "\nRandom Cards:\n>> ";
    for (int i = 0; i < 4; i++) {
        int temp = rand() % 13 + 1;
        // card.push_back(to_string(temp));
        switch (temp) {
            case 1: card.push_back("A"); break;
            case 11: card.push_back("J"); break;
            case 12: card.push_back("Q"); break;
            case 13: card.push_back("K"); break;
```



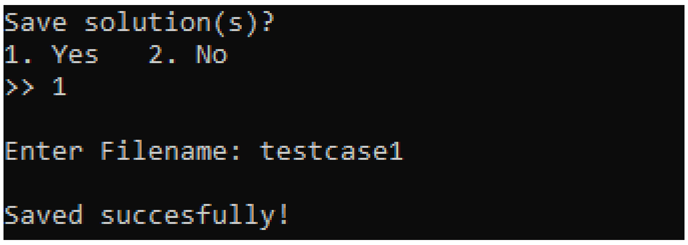
```cpp
            default: card.push_back(to_string(temp)); break;
        }
        cout << card[i] << " ";
    }
    cout << endl;
}

// Function to get a valid choice input from user (1/2)
void validateChoice(int &c) {
    string strIn;
    vector <string> in;
    cout << ">> ";
    getline(cin, strIn);
    const char* delim = " ";
    char *token = strtok(const_cast<char*>(strIn.c_str()), delim);
    while (token != nullptr) {
        in.push_back(string(token));
        token = strtok(nullptr, delim);
    }

    if (!(in.size() == 1 && (in[0] == "1" || in[0] == "2"))) {
        cout << "\nInvalid Input! Please enter [1/2]" << endl;
        in.clear();
        validateChoice(c);
        return;
    }

    c = stoi(in[0]);
}

// Function to save string result to txt file
void saveResult(string result, vector <string> card) {
    string fname;
    cout << "\nEnter Filename: ";
    cin >> fname;

    result = card[0] + " " + card[1] + " " + card[2] + " " + card[3] +
             "\n" + result;

    ofstream fout;
    fout.open("../test/" + fname + ".txt");
    fout << result;
    fout.close();
    cout << "\nSaved succesfully!\n";
}

// main program
int main() {
    vector <string> card;
    string result = "\n";
    int pil;

    cout << "\n24 GAME\n";
    cout << "\n1. Input Cards    2. Random Cards\n";
    validateChoice(pil);

    if (pil == 1) cardInput(card);
    else cardRandom(card);

    solve(card, result);
    cout << result;

    cout << "\nSave solution(s)?\n";
    cout << "1. Yes    2. No\n";
    validateChoice(pil);

    if (pil == 1) saveResult(result, card);

    return 0;
}
```

C. Input Output

1. Testcase 1 (K Q J A)

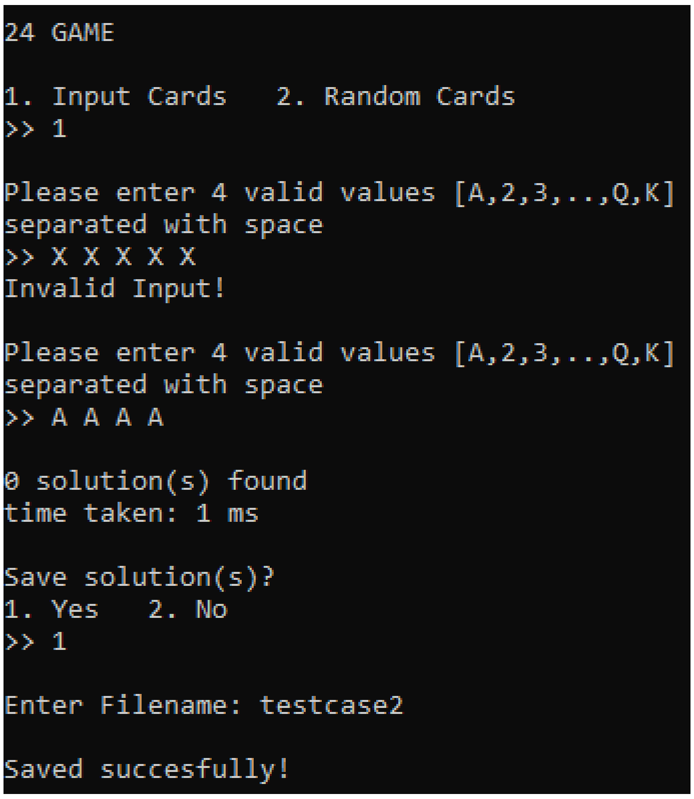


```
24 GAME

1. Input Cards   2. Random Cards
>> 1

Please enter 4 valid values [A,2,3,..,Q,K]
separated with space
>> K Q J A

32 solution(s) found
1 * (12 * (13 - 11))
1 * ((13 - 11) * 12)
(12 * 1) * (13 - 11)
12 * (1 * (13 - 11))
(1 * 12) * (13 - 11)
((1 * 13) - 11) * 12
12 * ((1 * 13) - 11)
(12 / 1) * (13 - 11)
12 / (1 / (13 - 11))
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
(13 - (1 * 11)) * 12
(12 * (13 - 11)) * 1
((13 * 1) - 11) * 12
((13 / 1) - 11) * 12
12 * (13 - (11 / 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)
(1 * (13 - 11)) * 12
(13 - 11) * (12 * 1)
((13 - 11) * 12) * 1
(13 - 11) * (12 / 1)
((13 - 11) * 12) / 1
time taken: 1 ms
```



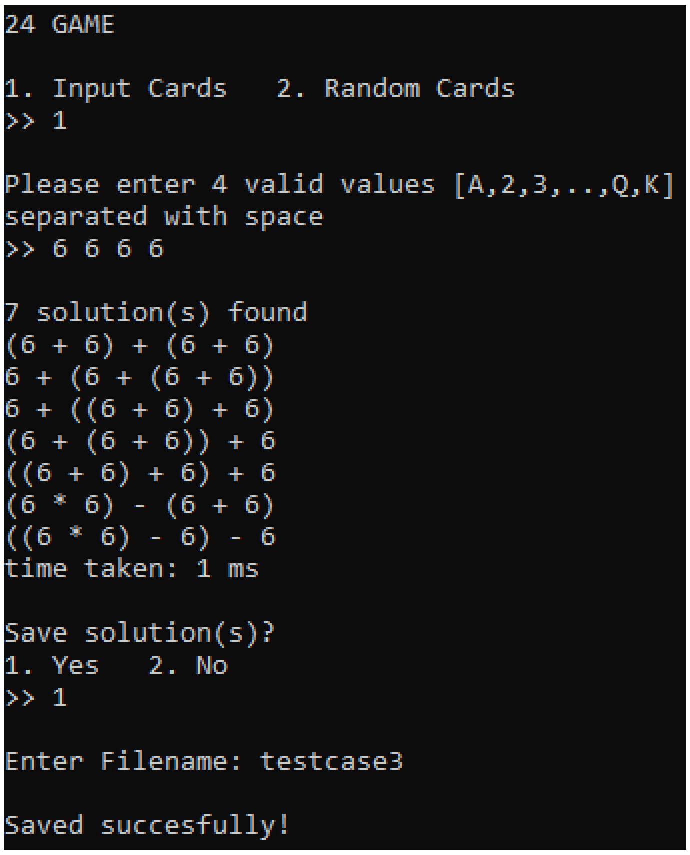
```
Save solution(s)?
1. Yes   2. No
>> 1

Enter Filename: testcase1

Saved succesfully!
```

2. Testcase 2 (A A A A) dengan input tidak valid



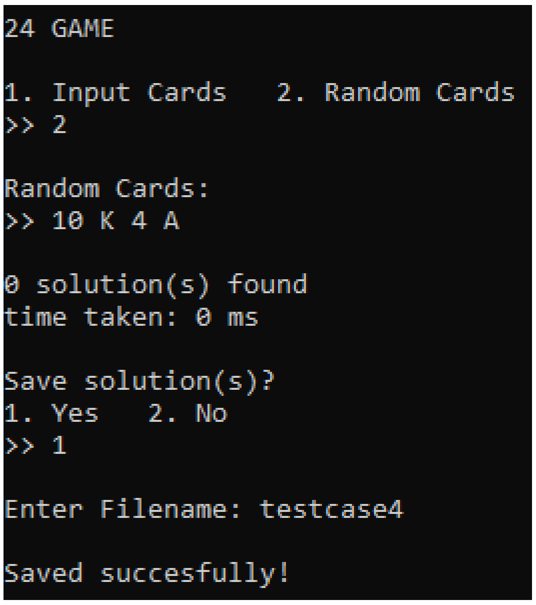
```
24 GAME

1. Input Cards   2. Random Cards
>> 1

Please enter 4 valid values [A,2,3,..,Q,K]
separated with space
>> X X X X X
Invalid Input!

Please enter 4 valid values [A,2,3,..,Q,K]
separated with space
>> A A A A

0 solution(s) found
time taken: 1 ms

Save solution(s)?
1. Yes   2. No
>> 1

Enter Filename: testcase2

Saved succesfully!
```

3. Textcase 3 (6 6 6 6)



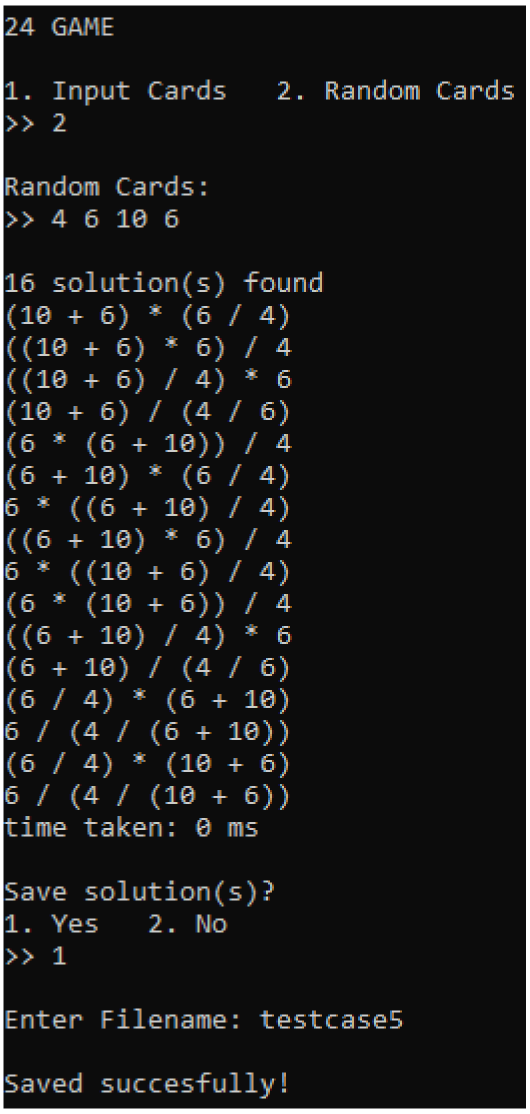
```
24 GAME

1. Input Cards   2. Random Cards
>> 1

Please enter 4 valid values [A,2,3,..,Q,K]
separated with space
>> 6 6 6 6

7 solution(s) found
(6 + 6) + (6 + 6)
6 + (6 + (6 + 6))
6 + ((6 + 6) + 6)
(6 + (6 + 6)) + 6
((6 + 6) + 6) + 6
(6 * 6) - (6 + 6)
((6 * 6) - 6) - 6
time taken: 1 ms

Save solution(s)?
1. Yes   2. No
>> 1

Enter Filename: testcase3

Saved succesfully!
```

4. Testcase 4 (Random)



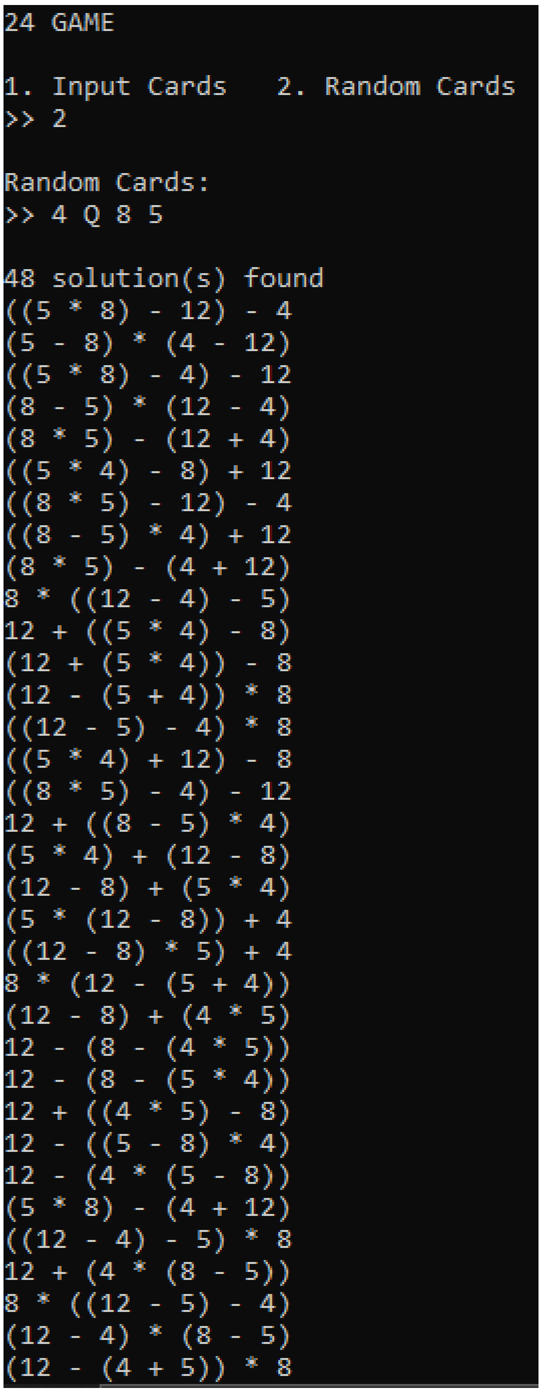
```
24 GAME

1. Input Cards   2. Random Cards
>> 2

Random Cards:
>> 10 K 4 A

0 solution(s) found
time taken: 0 ms

Save solution(s)?
1. Yes   2. No
>> 1

Enter Filename: testcase4

Saved succesfully!
```

5. Testcase 5 (Random)

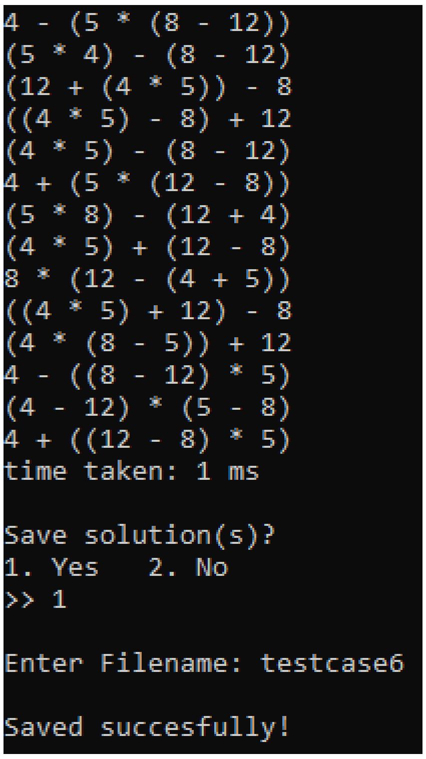


```
24 GAME

1. Input Cards   2. Random Cards
>> 2

Random Cards:
>> 4 6 10 6

16 solution(s) found
(10 + 6) * (6 / 4)
((10 + 6) * 6) / 4
((10 + 6) / 4) * 6
(10 + 6) / (4 / 6)
(6 * (6 + 10)) / 4
(6 + 10) * (6 / 4)
6 * ((6 + 10) / 4)
((6 + 10) * 6) / 4
6 * ((10 + 6) / 4)
(6 * (10 + 6)) / 4
((6 + 10) / 4) * 6
(6 + 10) / (4 / 6)
(6 / 4) * (6 + 10)
6 / (4 / (6 + 10))
(6 / 4) * (10 + 6)
6 / (4 / (10 + 6))
time taken: 0 ms

Save solution(s)?
1. Yes   2. No
>> 1

Enter Filename: testcase5

Saved succesfully!
```

6. Testcase 6 (Random)



```
24 GAME

1. Input Cards   2. Random Cards
>> 2

Random Cards:
>> 4 Q 8 5

48 solution(s) found
((5 * 8) - 12) - 4
(5 - 8) * (4 - 12)
((5 * 8) - 4) - 12
(8 - 5) * (12 - 4)
(8 * 5) - (12 + 4)
((5 * 4) - 8) + 12
((8 * 5) - 12) - 4
((8 - 5) * 4) + 12
(8 * 5) - (4 + 12)
8 * ((12 - 4) - 5)
12 + ((5 * 4) - 8)
(12 + (5 * 4)) - 8
(12 - (5 + 4)) * 8
((12 - 5) - 4) * 8
((5 * 4) + 12) - 8
((8 * 5) - 4) - 12
12 + ((8 - 5) * 4)
(5 * 4) + (12 - 8)
(12 - 8) + (5 * 4)
(5 * (12 - 8)) + 4
((12 - 8) * 5) + 4
8 * (12 - (5 + 4))
(12 - 8) + (4 * 5)
12 - (8 - (4 * 5))
12 - (8 - (5 * 4))
12 + ((4 * 5) - 8)
12 - ((5 - 8) * 4)
12 - (4 * (5 - 8))
(5 * 8) - (4 + 12)
((12 - 4) - 5) * 8
12 + (4 * (8 - 5))
8 * ((12 - 5) - 4)
(12 - 4) * (8 - 5)
(12 - (4 + 5)) * 8
```



```
4 - (5 * (8 - 12))
(5 * 4) - (8 - 12)
(12 + (4 * 5)) - 8
((4 * 5) - 8) + 12
(4 * 5) - (8 - 12)
4 + (5 * (12 - 8))
(5 * 8) - (12 + 4)
(4 * 5) + (12 - 8)
8 * (12 - (4 + 5))
((4 * 5) + 12) - 8
(4 * (8 - 5)) + 12
4 - ((8 - 12) * 5)
(4 - 12) * (5 - 8)
4 + ((12 - 8) * 5)
time taken: 1 ms

Save solution(s)?
1. Yes   2. No
>> 1

Enter Filename: testcase6

Saved succesfully!
```

D. Lampiran

Link to repository: bagas003/Tucil1_13521081

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	✓	