LAPORAN TUGAS KECIL I

IF2211 STRATEGI ALGORITMA

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun oleh:

Kenneth Ezekiel Suprantoni   13521089

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

Daftar Isi

BAGIAN I
ALGORITMA BRUTE FORCE

Algoritma *Brute Force*, adalah algoritma dengan pendekatan yang *straightforward* untuk memecahkan suatu masalah. Algoritma ini biasanya bergantung pada kekuatan komputasi yang tinggi untuk mendapatkan semua solusi yang tepat daripada menggunakan teknik yang canggih. Dalam penyelesaian permainan kartu 24 dengan pendekatan *brute force*, algoritma yang digunakan adalah sebagai berikut:

- misalkan a b c d sebagai 4 kartu/angka yang dipilih
- lakukan permutasi dari 4 kartu jika dipilih 2 kartu, sehingga didapatkan 3 angka (mis: (a .. b), sudah menjadi 1 angka), lalu operasikan (4 operasi)
- untuk semua permutasi * operasi tersebut, lakukan permutasi kembali dari 3 angka jika dipilih 2 angka, lalu operasikan (4 operasi) kembali (mis: ((a .. b) .. c) atau (c .. d))
- sisa 2 angka yang terakhir akan menentukan hasil akhir adalah 24 atau tidak jika dioperasikan (4 operasi)

Setelah algoritma menemukan hasil yang berjumlah 24, urutan pemilihan angka dan operasi yang dilakukan untuk mendapatkan angka 24 tersebut dicatat, sampai semua kemungkinan kombinasi sudah dicoba, dan jika ada suatu kombinasi yang sudah tercatat, tidak perlu dicatat kembali untuk meminimalisasi duplikat.

Secara algoritma, kompleksitas waktu yang ditawarkan bertumbuh dengan cepat ($O(n^3)$), tetapi karena n = 4, jumlah pengulangan dapat diperkirakan sebanyak 4*3*((4*2*(3)*4*4) + (2*4*(2)*4*(2)*4)) = 10752 pengulangan/kombinasi (sudah termasuk permutasi dari angka dan operator).

Algoritma yang digunakan mencoba semua kemungkinan solusi, sehingga semua solusi komutatif seperti a + b dan b + a akan dianggap dua solusi berbeda jika a != b, demikian pula untuk a * b dan b * a, dan juga a * (b .. c) dan (b .. c) * a.

BAB II
SOURCE PROGRAM

Projek ini ditulis dalam Bahasa C++, menggunakan *library*:

1. iostream (c++)
2. vector (c++)
3. string (c++)
4. fstream (c++)
5. sstream (c++)
6. cstdlib (c++)
7. ctime (c++)
8. chrono (c++)

Di dalam file *main.cpp*, modul-modul fungsi dibagi menjadi dua kategori, *miscellaneous modules* dan *algorithm*. *Miscellaneous modules* meliputi:

- checkCard
- translateCards
- printCards
- printList
- generateRandom
- getInput

Algorithm meliputi:

- op
- stringbuilder
- stringbuilderfromstring (2 overload)
- stringbuilderfromtwostring
- stringChecker
- solver

Berikut *source code*-nya:

```
// Miscellaneous

bool checkCard(string card)
{
    vector<string> listofcard = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};
    for (auto &x : listofcard)
    {
        if (card == x)
        {
            return true;
        }
    }
    return false;
```

```
}

void translateCards(vector<double> *input, const vector<string> cards)
{
    // LOCAL DICTIONARY

    // ALGORITHM
    for (auto &x : cards)
    {
        if (x == "2")
        {
            input->push_back(2);
        }
        else if (x == "3")
        {
            input->push_back(3);
        }
        else if (x == "4")
        {
            input->push_back(4);
        }
        else if (x == "5")
        {
            input->push_back(5);
        }
        else if (x == "6")
        {
            input->push_back(6);
        }
        else if (x == "7")
        {
            input->push_back(7);
        }
        else if (x == "8")
        {
            input->push_back(8);
        }
        else if (x == "9")
        {
            input->push_back(9);
        }
        else if (x == "10")
        {
            input->push_back(10);
        }
        else if (x == "J")
        {
            input->push_back(11);
        }
        else if (x == "Q")
        {
            input->push_back(12);
        }
        else if (x == "K")
        {
            input->push_back(13);
        }
        else if (x == "A")
        {
            input->push_back(1);
        }
    }
}

void printCards(const vector<string> cards)
{
    // LOCAL DICTIONARY
    int i;
    // ALGORITHM
    cout << "Cards: ";
    for (auto &x : cards)
    {
        std::cout << x << " ";
    }
    std::cout << endl;
}

void printList(const vector<double> list)
{
    // LOCAL DICTIONARY
    int i;
    // ALGORITHM
    cout << "List: ";
    for (i = 0; i < list.size(); i++)
    {
        std::cout << list.at(i) << " ";
    }
    std::cout << endl;
}

vector<double> generateRandom()
{
    // LOCAL DICTIONARY
    vector<double> output;
    double d;
    srand(time(NULL));
    // ALGORITHM

    for (int i = 0; i < 4; i++)
    {
        d = (rand() % 13) + 1;
        output.push_back(d);
    }
    return output;
}

void getInput(vector<double> *input)
{
    // LOCAL DICTIONARY
    string ans;
    string c, c1, c2, c3, c4;
    vector<string> cards;

    ifstream fin;

    string line;
    string temp;

    bool condition = true;

    int count;

    // ALGORITHM
    // input selection
    std::cout << "k : keyboard f : file r : random" << endl;
    std::cout << "Input? (k/f/r) (default: r) " << endl;
    getline(cin, ans);

    // from keyboard
    if (ans == "k")
    {
        // loop until valid
        while (condition)
        {
            std::cout << "Enter 4 cards: ";
            getline(cin, c);

            stringstream cc(c);

            string ctemp;

            int cnt = 0;
            cards.clear();
            while (cnt < 4 && cc >> ctemp)
            {
                if (checkCard(ctemp))
                {
                    // input is put to a vector
                    cards.push_back(ctemp);
                }
                else
                {
                    break;
                }
                cnt++;
            }

            if (cnt == 4)
            {
                condition = false;
            }
            else
            {
                std::cout << "Masukan tidak valid, harap diulangi!" << endl;
            }
        }

        // std::cout << c1 << " " << c2 << " " << c3 << " " << c4 << endl;
        printCards(cards);
        // translate cards to integer
        translateCards(input, cards);
    }
    // from file
    else if (ans == "f")
    {
        // open file
        fin.open("src/input.txt");
        // get line
        getline(fin, line);
        // std::cout << "here's the line: " << line << endl;
        fin.close();

        stringstream ss(line);
        count = 0;
        // check if the input is valid, if valid continue to put the input inside a vector, if not,
exit, and only takes 4 input
        while (ss >> temp && condition && count <= 3)
        {
            if (!checkCard(temp))
            {
                std::cout << "Masukan tidak valid, ganti isi file!" << endl;
                condition = false;
            }
            cards.push_back(temp);
            count++;
        }
        if (condition)
        {
            printCards(cards);
            // translates card
            translateCards(input, cards);
        }
    }
    else
    {
        *input = generateRandom();
        printList(*input);
    }
}
```

```
// Algorithms

double op(double num1, double num2, int opcode)
{
    if (opcode == 0)
    {
        return num1 + num2;
    }
    else if (opcode == 1)
    {
        return num1 - num2;
    }
    else if (opcode == 2)
    {
        return num1 * num2;
    }
    else if (opcode == 3)
    {
        if (num2 != 0)
        {
            return num1 / num2;
        }
        else
        {
            return -999;
        }
    }
    else
    {
        return -999;
    }
}

string stringbuilder(double num1, double num2, int opcode)
{
    // LOCAL DICTIONARY
    string solution = "(";
    int numtemp1 = num1;
    int numtemp2 = num2;
    // ALGORITHM
    solution += to_string(numtemp1);
    switch (opcode)
    {
    case 0:
        solution += " + ";
        break;
    case 1:
        solution += " - ";
        break;
    case 2:
        solution += " * ";
        break;
    case 3:
        solution += " / ";
        break;
    }
    solution += to_string(numtemp2);
    solution += ")";
    return solution;
}

string stringbuilderfromstring(string str1, double num2, int opcode)
{
    // LOCAL DICTIONARY
    string solution = "(";
    // int numtemp1 = num1;
    int numtemp2 = num2;
    // ALGORITHM
    solution += str1;
    switch (opcode)
    {
    case 0:
        solution += " + ";
        break;
    case 1:
        solution += " - ";
        break;
    case 2:
        solution += " * ";
        break;
    case 3:
        solution += " / ";
        break;
    }
    solution += to_string(numtemp2);
    solution += ")";
    return solution;
}

string stringbuilderfromstring(double num1, string str2, int opcode)
{
    // LOCAL DICTIONARY
    string solution = "(";
    int numtemp1 = num1;
    // int numtemp2 = num2;
    // ALGORITHM
    solution += to_string(numtemp1);
    switch (opcode)
    {
    case 0:
        solution += " + ";
        break;
    case 1:
        solution += " - ";
        break;
    case 2:
        solution += " * ";
        break;
    case 3:
        solution += " / ";
        break;
    }
    solution += str2;
    solution += ")";
    return solution;
}

string stringbuilderfromtwostring(string str1, string str2, int opcode)
{
    // LOCAL DICTIONARY
    string solution = "(";
    // ALGORITHM
    solution += str1;
    switch (opcode)
    {
    case 0:
        solution += " + ";
        break;
    case 1:
        solution += " - ";
        break;
    case 2:
        solution += " * ";
        break;
    case 3:
        solution += " / ";
        break;
    }
    solution += str2;
    solution += ")";
    return solution;
}

bool stringChecker(vector<string> list, string str)
{
    // LOCAL DICTIONARY

    // ALGORITHM
    // if (find(list.begin(), list.end(), str) != list.end()) {

    // }
    for (auto &x : list)
    {
        if (x == str)
        {
            return false;
        }
    }
    return true;
}

void solver(vector<double> list)
{
    // LOCAL DICTIONARY
    bool condition = true;

    int i, j, k, l, m, n, o, p, q, r, s, t, u;

    vector<string> solutions;

    vector<double> temp1, temp2, local;

    double num1, num2, num3, num4;
    double hasil1, hasil2, done;

    string solution;
```

```cpp
    char save;

    string filename;

    // const clock_t begintime = clock();

    // ALGORITHM
    local = list;
    auto t_start = chrono::high_resolution_clock::now();
    // pick card 1 from the list
    for (i = 0; i < 4; i++)
    {
        // std::cout << "i: " << i << endl;
        num1 = local.front();
        local.erase(local.begin());
        temp1 = local;
        // pick card 2 from the list
        for (j = 0; j < 3; j++)
        {
            // std::cout << "j: " << j << endl;
            num2 = temp1.front();
            temp1.erase(temp1.begin());
            temp2 = temp1;
            // operation on card 1 and card 2
            for (k = 0; k < 4; k++)
            {
                // std::cout << "k: " << k << endl;
                hasil1 = op(num1, num2, k);
                // std::cout << num1 << " " << k << " " << num2 << " = " << hasil1 << endl;
                // pick card 3 from the list
                for (l = 0; l < 2; l++)
                {
                    // std::cout << "l: " << l << endl;
                    num3 = temp2.front();
                    temp2.erase(temp2.begin());
                    // operation on (card 1 card 2) and card 3
                    for (m = 0; m < 4; m++)
                    {
                        // std::cout << "m: " << m << endl;
                        hasil2 = op(hasil1, num3, m);
                        // pick card 4 (last card on list)
                        num4 = temp2.front();
                        // operation on ((card 1 card 2) card 3) and card 4
                        for (n = 0; n < 4; n++)
                        {
                            // std::cout << "n: " << n << endl;
                            done = op(hasil2, num4, n);
                            if (done == 24)
                            {
                                solution = stringbuilder(num1, num2, k);
                                solution = stringbuilderfromstring(solution, num3, m);
                                solution = stringbuilderfromstring(solution, num4, n);
                                if (stringChecker(solutions, solution))
                                {
                                    // solution += " = ";
                                    // solution += to_string(done);
                                    solutions.push_back(solution);
                                }
                            }
                        }
                        // can be commented out
                        // operation on card 4 and ((card 1 card 2) card 3)
                        for (n = 0; n < 4; n++)
                        {
                            // std::cout << "n: " << n << endl;
                            done = op(num4, hasil2, n);
                            if (done == 24)
                            {
                                solution = stringbuilder(num1, num2, k);
                                solution = stringbuilderfromstring(solution, num3, m);
                                solution = stringbuilderfromstring(num4, solution, n);
                                if (stringChecker(solutions, solution))
                                {
                                    // solution += " = ";
                                    // solution += to_string(done);
                                    solutions.push_back(solution);
                                }
                            }
                        }
                    }
                    // operation on card 3 and (card 1 card 2)
                    for (m = 0; m < 4; m++)
                    {
                        // std::cout << "m: " << m << endl;
                        hasil2 = op(num3, hasil1, m);
                        // pick card 4 (last card on list)
                        num4 = temp2.front();
                        // operation on (card 3 (card 1 card 2)) and card 4
                        for (n = 0; n < 4; n++)
                        {
                            // std::cout << "n: " << n << endl;
                            done = op(hasil2, num4, n);
                            if (done == 24)
                            {
                                solution = stringbuilder(num1, num2, k);
                                solution = stringbuilderfromstring(num3, solution, m);
                                solution = stringbuilderfromstring(solution, num4, n);
                                if (stringChecker(solutions, solution))
                                {
                                    // solution += " = ";
                                    // solution += to_string(done);
                                    solutions.push_back(solution);
                                }
                            }
                        }
                        // can be commented out
                        // operation on card 4 and (card 3 (card 1 card 2))
                        for (n = 0; n < 4; n++)
                        {
                            // std::cout << "n: " << n << endl;
                            done = op(num4, hasil2, n);
                            if (done == 24)
                            {
                                solution = stringbuilder(num1, num2, k);
                                solution = stringbuilderfromstring(num3, solution, m);
                                solution = stringbuilderfromstring(num4, solution, n);
                                if (stringChecker(solutions, solution))
                                {
                                    // solution += " = ";
                                    // solution += to_string(done);
                                    // std::cout << num4 << " " << n << " " << hasil2 << " " << num1
<< k << num2 << m << num3 << endl;
                                    // std::cout << solution << endl;
                                    solutions.push_back(solution);
                                }
                            }
                        }
                    }
                    // pick card 4 (last card in list)
```

```
            num4 = temp2.front();
            // operation on card 3 and card 4 first
            for (o = 0; o < 4; o++)
            {
                // std::cout << "m: " << m << endl;
                hasil2 = op(num3, num4, o);
                // operation on (card 1 card 2) and (card 3 card 4)
                for (p = 0; p < 4; p++)
                {
                    // std::cout << "n: " << n << endl;

                    done = op(hasil1, hasil2, p);
                    if (done == -999)
                    {
                        continue;
                    }
                    if (done == 24)
                    {
                        solution = stringbuilder(num1, num2, k);
                        // std::cout << "solution temp: " << solution << endl;
                        string solutiontemp = stringbuilder(num3, num4, o);
                        solution = stringbuilderfromtwostring(solution, solutiontemp, p);
                        if (stringChecker(solutions, solution))
                        {
                            // solution += " = ";
                            // solution += to_string(done);
                            solutions.push_back(solution);
                        }
                    }
                }
            }
            temp2.push_back(num3);
        }
    }

    // pick card 3 first before operation
    for (q = 0; q < 2; q++)
    {
        num3 = temp2.front();
        temp2.erase(temp2.begin());
        // card 4 is the last card on the list
        num4 = temp2.front();
        // operation on card 2 and card 3
```

```
                for (r = 0; r < 4; r++)
                {
                    hasil1 = op(num2, num3, r);
                    // operation on card 1 and (card 2 card 3)
                    for (s = 0; s < 4; s++)
                    {
                        hasil2 = op(num1, hasil1, s);
                        // operation on (card 1 (card 2 card 3)) and card 4
                        for (t = 0; t < 4; t++)
                        {
                            done = op(hasil2, num4, t);
                            if (done == -999)
                            {
                                continue;
                            }
                            if (done == 24)
                            {
                                solution = stringbuilder(num2, num3, r);
                                // std::cout << "solution temp: " << solution << endl;
                                solution = stringbuilderfromstring(num1, solution, s);
                                solution = stringbuilderfromstring(solution, num4, t);
                                if (stringChecker(solutions, solution))
                                {
                                    // solution += " = ";
                                    // solution += to_string(done);
                                    solutions.push_back(solution);
                                } // (num1 .. num2) .. (num3 .. num4)
                            }
                        }
                        // can be commented out
                        // operation on card 4 and (card 1 (card 2 card 3))
                        for (t = 0; t < 4; t++)
                        {
                            done = op(num4, hasil2, t);
                            if (done == -999)
                            {
                                continue;
                            }
                            if (done == 24)
                            {

                                solution = stringbuilder(num2, num3, r);
                                // std::cout << "solution temp: " << solution << endl;
                                solution = stringbuilderfromstring(num1, solution, s);
                                solution = stringbuilderfromstring(num4, solution, t);
                                if (stringChecker(solutions, solution))
                                {
                                    // solution += " = ";
                                    // solution += to_string(done);
                                    solutions.push_back(solution);
                                }
                            }
                        }
                    }

                    // operation on (card 2 card 3) and card 4
                    for (s = 0; s < 4; s++)
                    {
                        hasil2 = op(hasil1, num4, s);
                        // operation on card 1 and ((card 2 card 3) card 4)
                        for (t = 0; t < 4; t++)
                        {
                            done = op(num1, hasil2, t);

                            if (done == -999)
                            {
                                continue;
                            }
                            if (done == 24)
                            {
                                solution = stringbuilder(num2, num3, r);
                                // std::cout << "solution temp: " << solution << endl;
                                solution = stringbuilderfromstring(solution, num4, s);
                                solution = stringbuilderfromstring(num1, solution, t);
                                if (stringChecker(solutions, solution))
                                {
                                    // solution += " = ";
                                    // solution += to_string(done);
                                    solutions.push_back(solution);
                                }
                            }
                        }
                        // can be commented out
                        // operation on ((card 2 card 3) card 4) and card 1
                        for (t = 0; t < 4; t++)
                        {
```

```
                        done = op(hasil2, num1, t);
                        if (done == -999)
                        {
                            continue;
                        }
                        if (done == 24)
                        {
                            solution = stringbuilder(num2, num3, r);
                            // std::cout << "solution temp: " << solution << endl;
                            solution = stringbuilderfromstring(solution, num4, s);
                            solution = stringbuilderfromstring(solution, num1, t);
                            if (stringChecker(solutions, solution))
                            {
                                // solution += " = ";
                                // solution += to_string(done);
                                solutions.push_back(solution);
                            }
                        }
                    }
                }
            }

            temp2.push_back(num3);
        }

        temp1.push_back(num2);
    }

    local.push_back(num1);
}
// std::cout << "executed in " << float(clock() - begintime) << endl;
auto t_end = chrono::high_resolution_clock::now();
double elapsed_time_ms = chrono::duration<double, milli>(t_end - t_start).count();
std::cout << "executed in " << elapsed_time_ms << " ms" << endl;

std::cout << solutions.size() << " solutions found" << endl;

for (auto &x : solutions)
{
    std::cout << x << endl;
}

std::cout << "save it to a file? (y/n) " << endl;
```

```
cin >> save;

if (save == 'y')

{
    std::cout << "rename file? (y/n) " << endl;

    cin >> name;

    filename += "test/";

    if (name == 'y')
    {
        std::cout << "filename: ";
        cin >> nameoffile;
        filename += nameoffile;
    }
    else
    {
        for (auto &x : list)
        {
            if (x == 1)
            {
                filename += "A ";
            }
            else if (x == 11)
            {
                filename += "J ";
            }
            else if (x == 12)
            {
                filename += "Q ";
            }
            else if (x == 13)
            {
                filename += "K ";
            }
            else
            {
                int num = x;
                filename += to_string(num);
                filename += " ";
                }
            }
        }

        filename += ".txt";
        ofstream fileout(filename);

        for (auto &x : solutions)
        {
            fileout << x;
            fileout << "\n";
        }

        for (auto &x : list)
        {
            if (x == 1)
            {
                fileout << "A ";
            }
            else if (x == 11)
            {
                fileout << "J ";
            }
            else if (x == 12)
            {
                fileout << "Q ";
            }
            else if (x == 13)
            {
                fileout << "K ";
            }
            else
            {
                int num = x;
                fileout << to_string(num);
                fileout << " ";
            }
        }
        fileout << "\n";

        fileout << solutions.size();

        fileout << " solutions";
        fileout.close();
    }
}
```

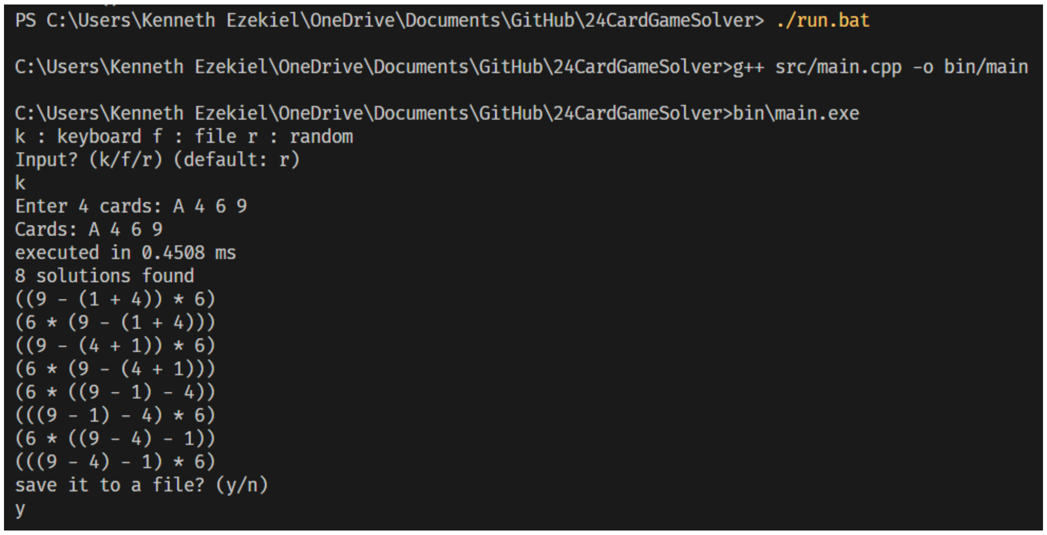
```
int main()
{
    vector<double> ans;
    getInput(&ans);
    // printList(ans);

    solver(ans);

    return 0;
}
```

Selain itu, diimplementasikan juga sebuah script python sebagai *checker* dari hasil yang dikeluarkan oleh program dalam *script.py*:

```
import os
from pathlib import Path

print("--------------------------- CHECKER ---------------------------\n")
cwd = os.getcwd()
folder = 'test'
directory = os.path.join(cwd, folder)

for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    if os.path.isfile(f):
        print("file:", f)
        string = Path(f).read_text()

        arr = string.split("\n")
        del arr[-1]
        del arr[-1]
        cnt = 0
        err = []
        for line in arr:
            n = eval(line) == 24
            # print(n)
            if not n:
                cnt += 1
                err.append(line)


        set_arr = set(arr)
        print("length of unique strings:", len(set_arr))
        print("length of array of solutions:", len(arr))
        print("number of errors:", cnt, "\n")

        for i in err:
            print(i)
```

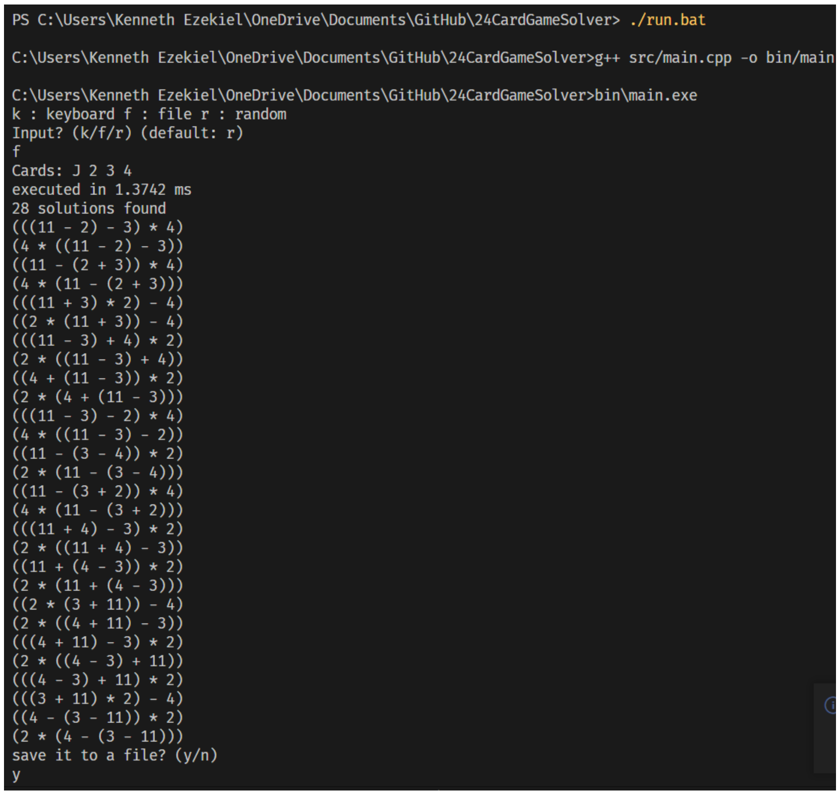
BAGIAN III
SCREENSHOT HASIL TEST

Input: A 4 6 9 (dari Keyboard)



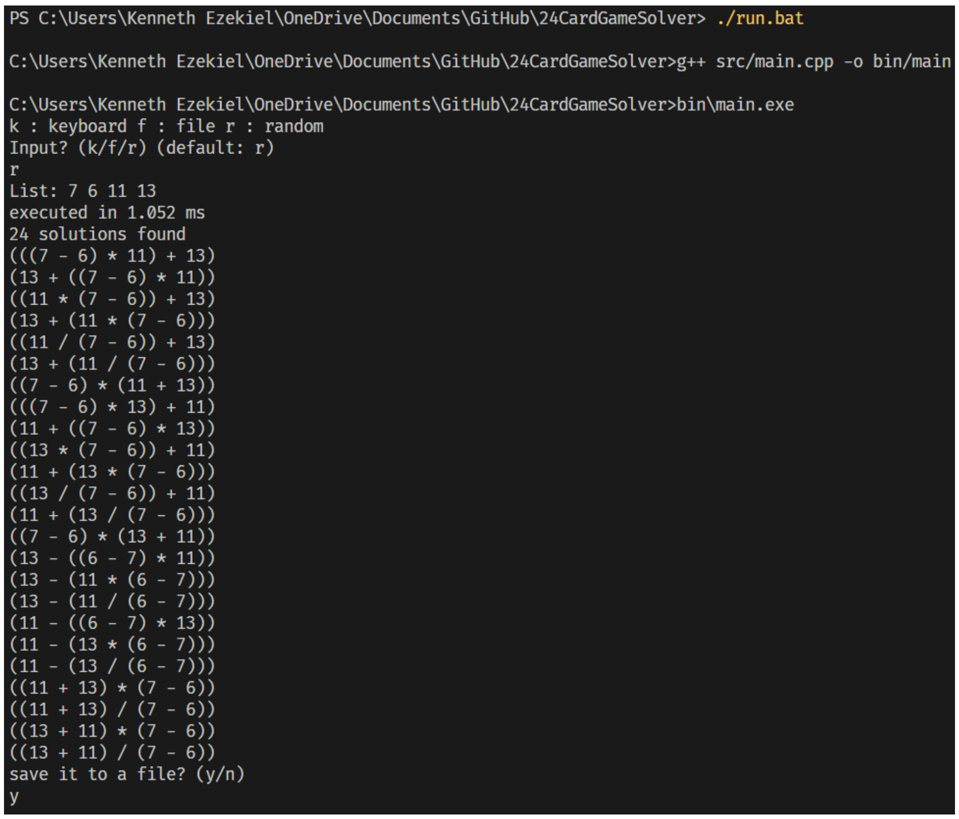
```
PS C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver> ./run.bat

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>g++ src/main.cpp -o bin/main

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>bin\main.exe
k : keyboard f : file r : random
Input? (k/f/r) (default: r)
k
Enter 4 cards: A 4 6 9
Cards: A 4 6 9
executed in 0.4508 ms
8 solutions found
((9 - (1 + 4)) * 6)
(6 * (9 - (1 + 4)))
((9 - (4 + 1)) * 6)
(6 * (9 - (4 + 1)))
(6 * ((9 - 1) - 4))
(((9 - 1) - 4) * 6)
(6 * ((9 - 4) - 1))
(((9 - 4) - 1) * 6)
save it to a file? (y/n)
y
```

Input: J 2 3 4 (dari File)



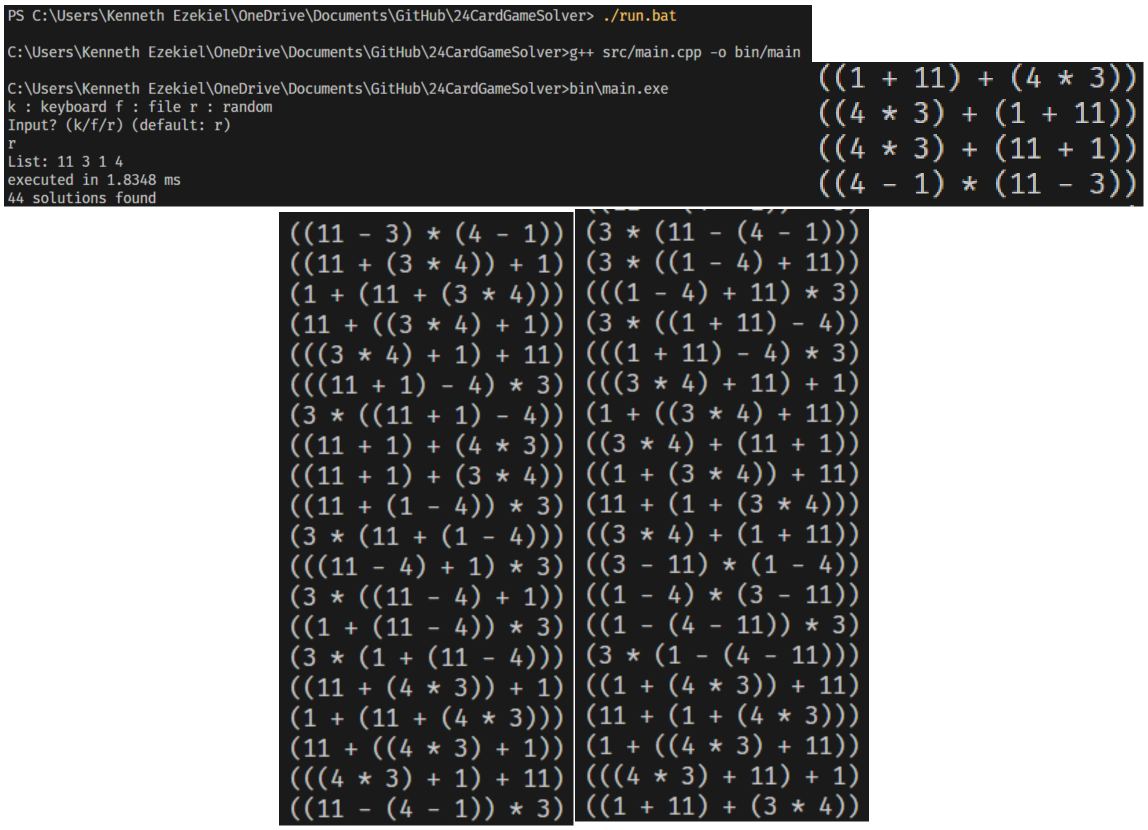
```
PS C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver> ./run.bat

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>g++ src/main.cpp -o bin/main

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>bin\main.exe
k : keyboard f : file r : random
Input? (k/f/r) (default: r)
f
Cards: J 2 3 4
executed in 1.3742 ms
28 solutions found
(((11 - 2) - 3) * 4)
(4 * ((11 - 2) - 3))
((11 - (2 + 3)) * 4)
(4 * (11 - (2 + 3)))
(((11 + 3) * 2) - 4)
((2 * (11 + 3)) - 4)
(((11 - 3) + 4) * 2)
(2 * ((11 - 3) + 4))
((4 + (11 - 3)) * 2)
(2 * (4 + (11 - 3)))
(((11 - 3) - 2) * 4)
(4 * ((11 - 3) - 2))
((11 - (3 - 4)) * 2)
(2 * (11 - (3 - 4)))
((11 - (3 + 2)) * 4)
(4 * (11 - (3 + 2)))
(((11 + 4) - 3) * 2)
(2 * ((11 + 4) - 3))
((11 + (4 - 3)) * 2)
(2 * (11 + (4 - 3)))
((2 * (3 + 11)) - 4)
(2 * ((4 + 11) - 3))
(((4 + 11) - 3) * 2)
(2 * ((4 - 3) + 11))
(((4 - 3) + 11) * 2)
(((3 + 11) * 2) - 4)
((4 - (3 - 11)) * 2)
(2 * (4 - (3 - 11)))
save it to a file? (y/n)
y
```

Input: 7 6 J K (Random)



```
PS C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver> ./run.bat

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>g++ src/main.cpp -o bin/main

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>bin\main.exe
k : keyboard f : file r : random
Input? (k/f/r) (default: r)
r
List: 7 6 11 13
executed in 1.052 ms
24 solutions found
(((7 - 6) * 11) + 13)
(13 + ((7 - 6) * 11))
((11 * (7 - 6)) + 13)
(13 + (11 * (7 - 6)))
((11 / (7 - 6)) + 13)
(13 + (11 / (7 - 6)))
((7 - 6) * (11 + 13))
(((7 - 6) * 13) + 11)
(11 + ((7 - 6) * 13))
((13 * (7 - 6)) + 11)
(11 + (13 * (7 - 6)))
((13 / (7 - 6)) + 11)
(11 + (13 / (7 - 6)))
((7 - 6) * (13 + 11))
(13 - ((6 - 7) * 11))
(13 - (11 * (6 - 7)))
(13 - (11 / (6 - 7)))
(11 - ((6 - 7) * 13))
(11 - (13 * (6 - 7)))
(11 - (13 / (6 - 7)))
((11 + 13) * (7 - 6))
((11 + 13) / (7 - 6))
((13 + 11) * (7 - 6))
((13 + 11) / (7 - 6))
save it to a file? (y/n)
y
```

Input: J 3 1 4 (Random)



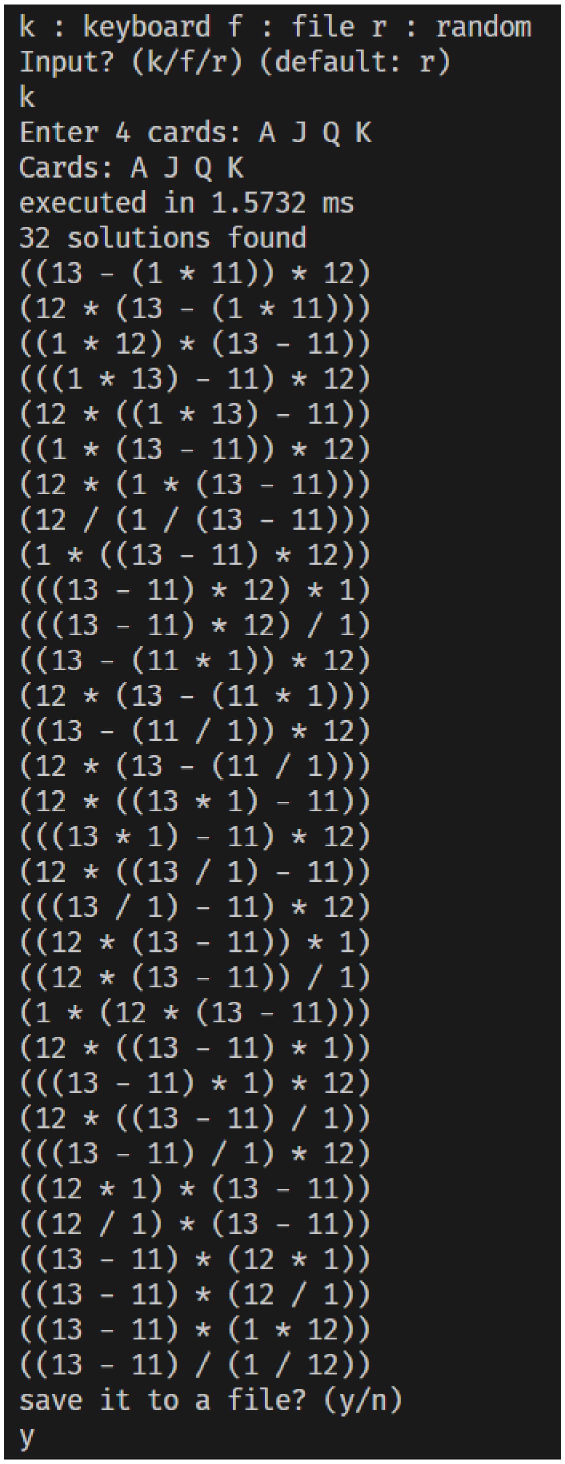
```
PS C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver> ./run.bat

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>g++ src/main.cpp -o bin/main

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>bin\main.exe
k : keyboard f : file r : random
Input? (k/f/r) (default: r)
r
List: 11 3 1 4
executed in 1.8348 ms
44 solutions found
```

```
((1 + 11) + (4 * 3))
((4 * 3) + (1 + 11))
((4 * 3) + (11 + 1))
((4 - 1) * (11 - 3))
```

```
((11 - 3) * (4 - 1))
((11 + (3 * 4)) + 1)
(1 + (11 + (3 * 4)))
(11 + ((3 * 4) + 1))
(((3 * 4) + 1) + 11)
(((11 + 1) - 4) * 3)
(3 * ((11 + 1) - 4))
((11 + 1) + (4 * 3))
((11 + 1) + (3 * 4))
((11 + (1 - 4)) * 3)
(3 * (11 + (1 - 4)))
(((11 - 4) + 1) * 3)
(3 * ((11 - 4) + 1))
((1 + (11 - 4)) * 3)
(3 * (1 + (11 - 4)))
((11 + (4 * 3)) + 1)
(1 + (11 + (4 * 3)))
(11 + ((4 * 3) + 1))
(((4 * 3) + 1) + 11)
((11 - (4 - 1)) * 3)
```

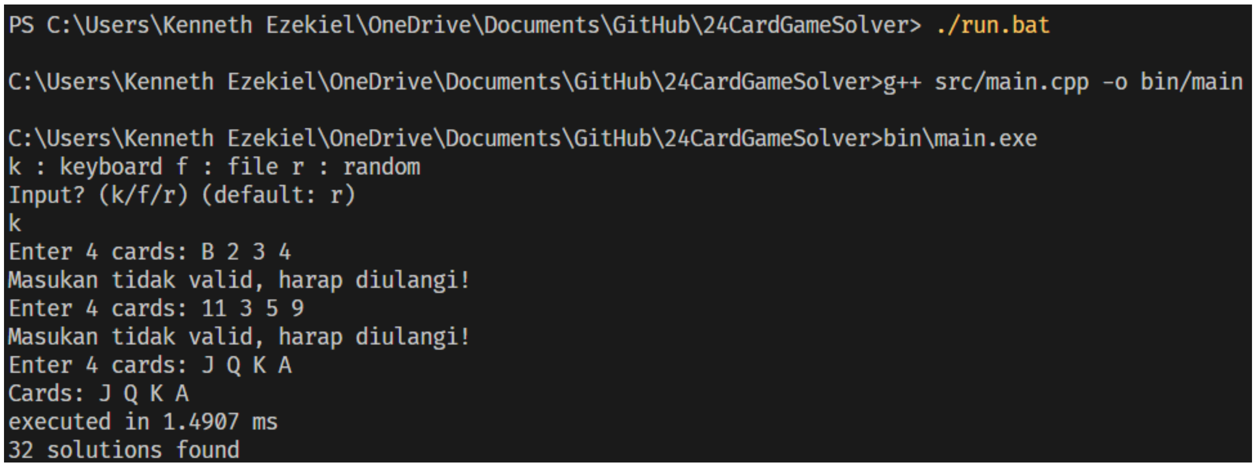
```
(3 * (11 - (4 - 1)))
(3 * ((1 - 4) + 11))
(((1 - 4) + 11) * 3)
(3 * ((1 + 11) - 4))
(((1 + 11) - 4) * 3)
(((3 * 4) + 11) + 1)
(1 + ((3 * 4) + 11))
((3 * 4) + (11 + 1))
((1 + (3 * 4)) + 11)
(11 + (1 + (3 * 4)))
((3 * 4) + (1 + 11))
((3 - 11) * (1 - 4))
((1 - 4) * (3 - 11))
((1 - (4 - 11)) * 3)
(3 * (1 - (4 - 11)))
((1 + (4 * 3)) + 11)
(11 + (1 + (4 * 3)))
(1 + ((4 * 3) + 11))
(((4 * 3) + 11) + 1)
((1 + 11) + (3 * 4))
```

Input: A J Q K (dari Keyboard)

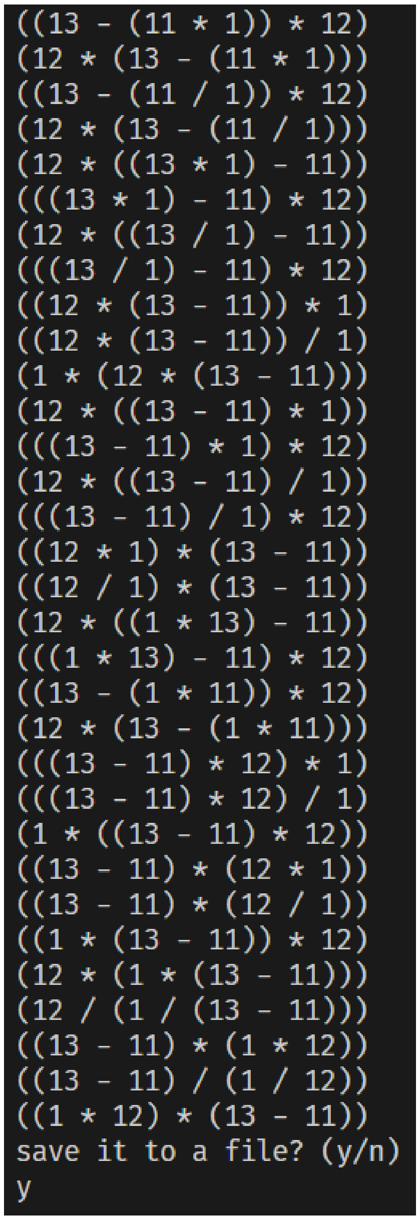


```
k : keyboard f : file r : random
Input? (k/f/r) (default: r)
k
Enter 4 cards: A J Q K
Cards: A J Q K
executed in 1.5732 ms
32 solutions found
((13 - (1 * 11)) * 12)
(12 * (13 - (1 * 11)))
((1 * 12) * (13 - 11))
(((1 * 13) - 11) * 12)
(12 * ((1 * 13) - 11))
((1 * (13 - 11)) * 12)
(12 * (1 * (13 - 11)))
(12 / (1 / (13 - 11)))
(1 * ((13 - 11) * 12))
(((13 - 11) * 12) * 1)
(((13 - 11) * 12) / 1)
((13 - (11 * 1)) * 12)
(12 * (13 - (11 * 1)))
((13 - (11 / 1)) * 12)
(12 * (13 - (11 / 1)))
(12 * ((13 * 1) - 11))
(((13 * 1) - 11) * 12)
(12 * ((13 / 1) - 11))
(((13 / 1) - 11) * 12)
((12 * (13 - 11)) * 1)
((12 * (13 - 11)) / 1)
(1 * (12 * (13 - 11)))
(12 * ((13 - 11) * 1))
(((13 - 11) * 1) * 12)
(12 * ((13 - 11) / 1))
(((13 - 11) / 1) * 12)
((12 * 1) * (13 - 11))
((12 / 1) * (13 - 11))
((13 - 11) * (12 * 1))
((13 - 11) * (12 / 1))
((13 - 11) * (1 * 12))
((13 - 11) / (1 / 12))
save it to a file? (y/n)
y
```

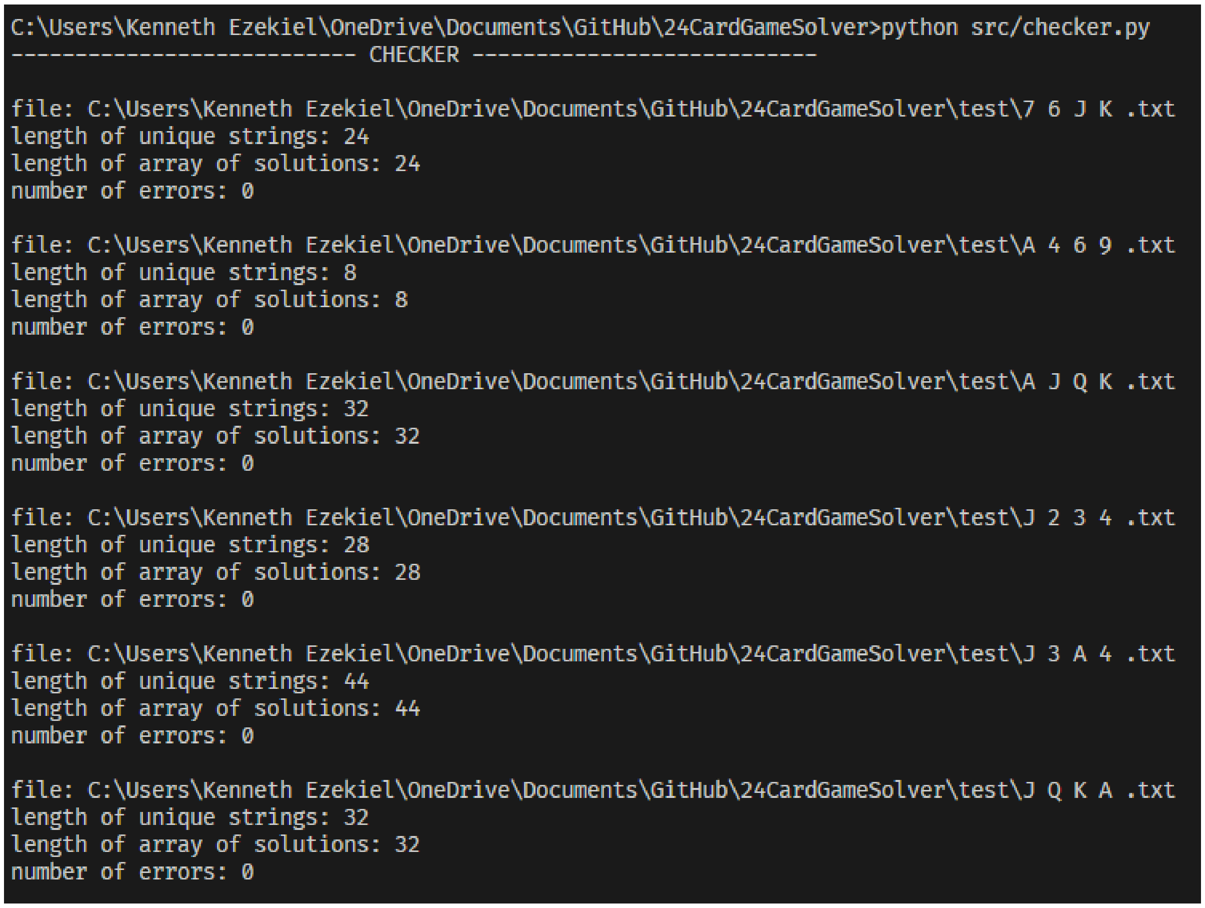
Input: test B 2 3 4, 11 3 5 9, valid J Q K A (test sama atau tidak jumlah solusinya dengan sebelumnya)



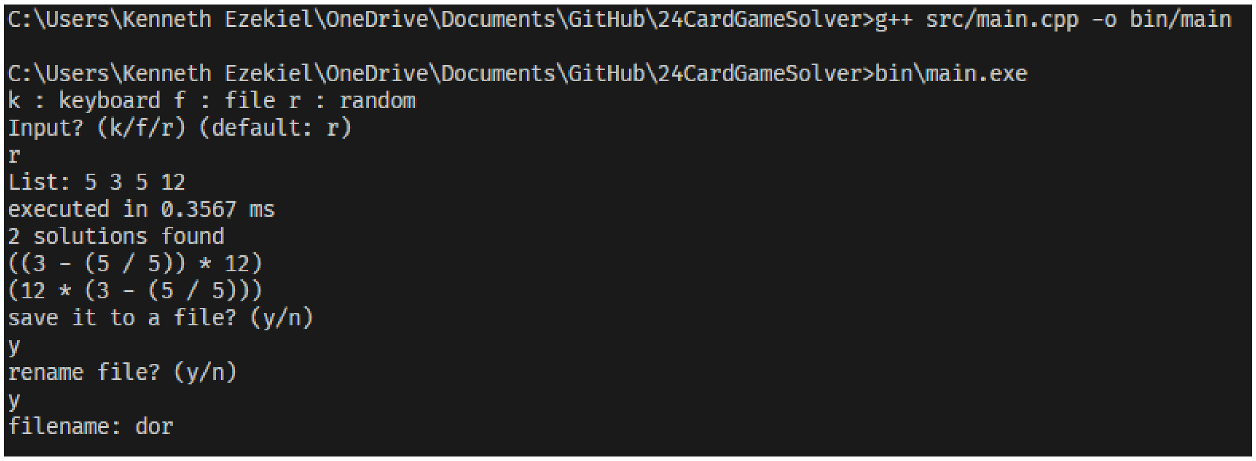
```
PS C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver> ./run.bat

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>g++ src/main.cpp -o bin/main

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>bin\main.exe
k : keyboard f : file r : random
Input? (k/f/r) (default: r)
k
Enter 4 cards: B 2 3 4
Masukan tidak valid, harap diulangi!
Enter 4 cards: 11 3 5 9
Masukan tidak valid, harap diulangi!
Enter 4 cards: J Q K A
Cards: J Q K A
executed in 1.4907 ms
32 solutions found
```



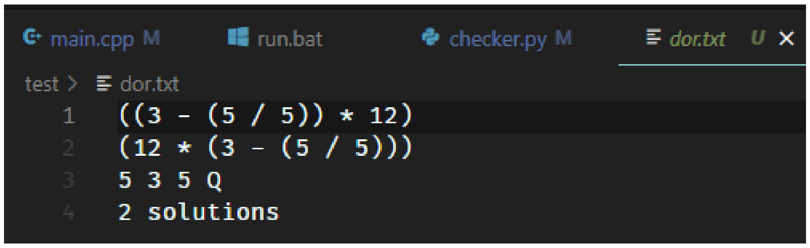
```
((13 - (11 * 1)) * 12)
(12 * (13 - (11 * 1)))
((13 - (11 / 1)) * 12)
(12 * (13 - (11 / 1)))
(12 * ((13 * 1) - 11))
(((13 * 1) - 11) * 12)
(12 * ((13 / 1) - 11))
(((13 / 1) - 11) * 12)
((12 * (13 - 11)) * 1)
((12 * (13 - 11)) / 1)
(1 * (12 * (13 - 11)))
(12 * ((13 - 11) * 1))
(((13 - 11) * 1) * 12)
(12 * ((13 - 11) / 1))
(((13 - 11) / 1) * 12)
((12 * 1) * (13 - 11))
((12 / 1) * (13 - 11))
(12 * ((1 * 13) - 11))
(((1 * 13) - 11) * 12)
((13 - (1 * 11)) * 12)
(12 * (13 - (1 * 11)))
(((13 - 11) * 12) * 1)
(((13 - 11) * 12) / 1)
(1 * ((13 - 11) * 12))
((13 - 11) * (12 * 1))
((13 - 11) * (12 / 1))
((1 * (13 - 11)) * 12)
(12 * (1 * (13 - 11)))
(12 / (1 / (13 - 11)))
((13 - 11) * (1 * 12))
((13 - 11) / (1 / 12))
((1 * 12) * (13 - 11))
save it to a file? (y/n)
y
```



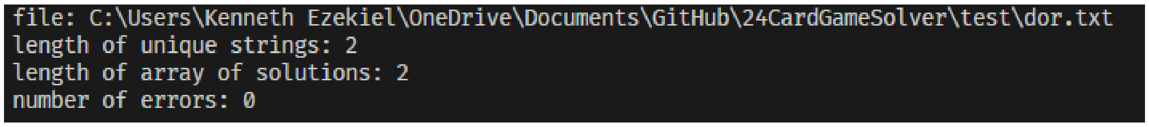
```
C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>python src/checker.py
--------------------------- CHECKER ---------------------------

file: C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver\test\7 6 J K .txt
length of unique strings: 24
length of array of solutions: 24
number of errors: 0

file: C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver\test\A 4 6 9 .txt
length of unique strings: 8
length of array of solutions: 8
number of errors: 0

file: C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver\test\A J Q K .txt
length of unique strings: 32
length of array of solutions: 32
number of errors: 0

file: C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver\test\J 2 3 4 .txt
length of unique strings: 28
length of array of solutions: 28
number of errors: 0

file: C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver\test\J 3 A 4 .txt
length of unique strings: 44
length of array of solutions: 44
number of errors: 0

file: C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver\test\J Q K A .txt
length of unique strings: 32
length of array of solutions: 32
number of errors: 0
```



```
C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>g++ src/main.cpp -o bin/main

C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver>bin\main.exe
k : keyboard f : file r : random
Input? (k/f/r) (default: r)
r
List: 5 3 5 12
executed in 0.3567 ms
2 solutions found
((3 - (5 / 5)) * 12)
(12 * (3 - (5 / 5)))
save it to a file? (y/n)
y
rename file? (y/n)
y
filename: dor
```



```
((3 - (5 / 5)) * 12)
(12 * (3 - (5 / 5)))
5 3 5 Q
2 solutions
```



```
file: C:\Users\Kenneth Ezekiel\OneDrive\Documents\GitHub\24CardGameSolver\test\dor.txt
length of unique strings: 2
length of array of solutions: 2
number of errors: 0
```

LINK REPOSITORY

https://github.com/KenEzekiel/Tucil1_13521089

CHECKLIST

Poin	Ya	Tidak
Program dapat 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	✓	

+ Program dapat menerima input dari sebuah file teks