Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*
Semester II tahun 2022/2023

Oleh:
Puti Nabilla Aidira (13521088)

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2022

Daftar Isi

Cek List Poin

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	✓	

Deskripsi Algoritma

Algoritma yang digunakan untuk menemukan solusi-solusi pada persoalan kartu 24 adalah algoritma Brute Force yang dijabarkan dalam langkah-langkah berikut.

1. Nomor-nomor kartu disimpan dalam variabel bertipe float a, b, c, dan d.
2. Keempat variabel a, b, c, dan d dievaluasi dalam fungsi `GROUPING1`, `GROUPING2`, `GROUPING3`, `GROUPING4`, dan `GROUPING5`, yang mengevaluasi operasi dengan urutan prioritas (tanda kurung) berturut-turut: `(a_b)_(c_d)`, `((a_b)_c)_d`, `(a_(b_c))_d`, `a_((b_c)_d)`, dan `a_(b_(c_d))`.
3. Pada setiap fungsi `GROUPING` dilakukan iterasi pada setiap permutasi nilai-nilai a, b, c, d (24 permutasi) dengan fungsi `PERMUTE`. Implementasi fungsi `PERMUTE` adalah dengan menggunakan switch-case untuk setiap permutasi (case 0-23) . Pada setiap case, nilai-nilai a, b, c, d di-*assign* dengan permutasinya.
4. Di dalam setiap iterasi permutasi nilai-nilai a, b, c, d, dilakukan pula iterasi sebanyak 4^3 untuk mempermutasikan kombinasi operasi (terdapat 4 jenis operator: +, -, *, / dan 3 kali operasi).
5. Selanjutnya, pada setiap 24 * 4^3 iterasi, operasi antara a, b, c, dan, d dihitung menggunakan fungsi `OPERATE` sesuai operator serta uratan prioritasnya.
6. Hasil operasi dibandingkan dengan 24 menggunakan fungsi `is24`. Jika hasil operasi berada pada rentang 24 ± 0.000001, hasil operasi tersebut dianggap sama dengan 24 dan menjadi salah satu solusi.
7. Setiap solusi yang memenuhi disimpan dengan format string ke dalam unsorted_set.
8. Pada akhir algoritma, semua solusi yang mungkin tersimpan pada unsorted_set.

Source Code

Program ditulis dalam bahasa pemograman C++. Berikut dilampirkan *source code* program.

```
#include <iostream>
#include <unordered_set>
#include <fstream>

using namespace std;

unordered_set <string> setres;

float OPERATE(char op, float val1, float val2){
   switch(op){
       case '+': return val1 + val2;
       case '-': return val1 - val2;
       case '*': return val1 * val2;
       case '/': return val1 / val2;
       default: return -1;
   }
}

float INPUT_CONVERTER(string s){
   if (s == "A" || s == "1") return 1;
   else if (s == "2") return 2;
   else if (s == "3") return 3;
   else if (s == "4") return 4;
   else if (s == "5") return 5;
   else if (s == "6") return 6;
   else if (s == "7") return 7;
   else if (s == "8") return 8;
   else if (s == "9") return 9;
   else if (s == "10") return 10;
   else if (s == "J") return 11;
   else if (s == "Q") return 12;
   else if (s == "K") return 13;
   else return -1;
}
```

```
bool is24(float x){
   return (24 - x <= 0.000001 && 24 - x >= -0.000001);
}

void PERMUTE(int i, float* a, float* b, float* c, float* d, float w, float x, float
y, float z){
   switch(i){
       case 0: break;
       case 1: {
           *a = x;
           *b = w;
           *c = y;
           *d = z;
           break;
       }
       case 2: {
           *a = y;
           *b = w;
           *c = x;
           *d = z;
           break;
       }
       case 3: {
           *a = w;
           *b = y;
           *c = x;
           *d = z;
           break;
       }
       case 4: {
           *a = x;
           *b = y;
           *c = w;
           *d = z;
           break;
       }
       case 5: {
           *a = y;
           *b = x;
           *c = w;
```

```
            *d = z;
            break;
        }
        case 6: {
            *a = y;
            *b = x;
            *c = z;
            *d = w;
            break;
        }
        case 7: {
            *a = x;
            *b = y;
            *c = z;
            *d = w;
            break;
        }
        case 8: {
            *a = z;
            *b = y;
            *c = x;
            *d = w;
            break;
        }
        case 9: {
            *a = y;
            *b = z;
            *c = x;
            *d = w;
            break;
        }
        case 10: {
            *a = x;
            *b = z;
            *c = y;
            *d = w;
            break;
        }
        case 11: {
            *a = z;
            *b = x;
```

```
            *c = y;
            *d = w;
            break;
        }
        case 12: {
            *a = z;
            *b = w;
            *c = y;
            *d = x;
            break;
        }
        case 13: {
            *a = w;
            *b = z;
            *c = y;
            *d = x;
            break;
        }
        case 14: {
            *a = y;
            *b = z;
            *c = w;
            *d = x;
            break;
        }
        case 15: {
            *a = z;
            *b = y;
            *c = w;
            *d = x;
            break;
        }
        case 16: {
            *a = w;
            *b = y;
            *c = z;
            *d = x;
            break;
        }
        case 17:
        {
```

```
            *a = y;
            *b = w;
            *c = z;
            *d = x;
            break;
        }
        case 18:
        {
            *a = x;
            *b = w;
            *c = z;
            *d = y;
            break;
        }
        case 19:
        {
            *a = w;
            *b = x;
            *c = z;
            *d = y;
            break;
        }
        case 20:
        {
            *a = z;
            *b = x;
            *c = w;
            *d = y;
            break;
        }
        case 21:
        {
            *a = x;
            *b = z;
            *c = w;
            *d = y;
            break;
        }
        case 22:
        {
            *a = w;
```

```
            *b = z;
            *c = x;
            *d = y;
            break;
        }
        case 23:
        {
            *a = z;
            *b = w;
            *c = x;
            *d = y;
            break;
        }
    }
}

void GROUPING1(float a, float b, float c, float d, char ops[4]){
    // (a_b)_(c_d)
    float w = a, x = b, y = c, z = d, res, res1, res2;
    for(int i = 0; i < 24; i++){
        PERMUTE(i, &a, &b, &c, &d, w, x, y, z);
        for(int j = 0; j < 4; j++){
            for(int k = 0; k < 4; k++){
                for(int l = 0; l < 4; l++){
                    res1 = OPERATE(ops[j], a, b);
                    res2 = OPERATE(ops[k], c, d);
                    res = OPERATE(ops[l], res1, res2);
                    if(is24(res)){
                        setres.insert("(" + to_string((int)a) + ops[j] +
to_string((int)b) + ")" + ops[l] + "(" + to_string((int)c) + ops[k] +
to_string((int)d) + ")");
                    }
                }
            }
        }
    }
}

void GROUPING2(float a, float b, float c, float d, char ops[4]){
    // ((a_b)_c)_d
    float w = a, x = b, y = c, z = d, res, res1, res2;
```

```
   for(int i = 0; i < 24; i++){
       PERMUTE(i, &a, &b, &c, &d, w, x, y, z);
       for(int j = 0; j < 4; j++){
           for(int k = 0; k < 4; k++){
               for(int l = 0; l < 4; l++){
                   res1 = OPERATE(ops[j], a, b);
                   res2 = OPERATE(ops[k], res1, c);
                   res = OPERATE(ops[l], res2, d);
                   if(is24(res)){
                       setres.insert("((" + to_string((int)a) + ops[j] +
to_string((int)b) + ")" + ops[k] + to_string((int)c) + ")" + ops[l] +
to_string((int)d));
                   }
               }
           }
       }
   }
}

void GROUPING3(float a, float b, float c, float d, char ops[4]){
   // (a_(b_c))_d
   float w = a, x = b, y = c, z = d, res, res1, res2;
   for(int i = 0; i < 24; i++){
       PERMUTE(i, &a, &b, &c, &d, w, x, y, z);
       for(int j = 0; j < 4; j++){
           for(int k = 0; k < 4; k++){
               for(int l = 0; l < 4; l++){
                   res1 = OPERATE(ops[j], b, c);
                   res2 = OPERATE(ops[k], a, res1);
                   res = OPERATE(ops[l], res2, d);
                   if(is24(res)){
                       setres.insert("(" + to_string((int)a) + ops[k] + "(" +
to_string((int)b) + ops[j] + to_string((int)c) + "))" + ops[l] + to_string((int)d));
                   }
               }
           }
       }
   }
}

void GROUPING4(float a, float b, float c, float d, char ops[4]){
```

```
    // a_((b_c)_d)
    float w = a, x = b, y = c, z = d, res, res1, res2;
    for(int i = 0; i < 24; i++){
        PERMUTE(i, &a, &b, &c, &d, w, x, y, z);
        for(int j = 0; j < 4; j++){
            for(int k = 0; k < 4; k++){
                for(int l = 0; l < 4; l++){
                    res1 = OPERATE(ops[j], b, c);
                    res2 = OPERATE(ops[k], res1, d);
                    res = OPERATE(ops[l], a, res2);
                    if(is24(res)){
                        setres.insert(to_string((int)a) + ops[l] + "((" +
to_string((int)b) + ops[j] + to_string((int)c) + ")" + ops[k] + to_string((int)d) +
"))");
                    }
                }
            }
        }
    }
}

void GROUPING5(float a, float b, float c, float d, char ops[4]){
    // a_(b_(c_d))
    float w = a, x = b, y = c, z = d, res, res1, res2;
    for(int i = 0; i < 24; i++){
        PERMUTE(i, &a, &b, &c, &d, w, x, y, z);
        for(int j = 0; j < 4; j++){
            for(int k = 0; k < 4; k++){
                for(int l = 0; l < 4; l++){
                    res1 = OPERATE(ops[j], c, d);
                    res2 = OPERATE(ops[k], b, res1);
                    res = OPERATE(ops[l], a, res2);
                    if(is24(res)){
                        setres.insert(to_string((int)a) + ops[l] + "(" +
to_string((int)b) + ops[k] + "(" + to_string((int)c) + ops[j] + to_string((int)d) +
"))");
                    }
                }
            }
        }
    }
```

```
}

void SPLASHSCREEN(){
   cout << "
_____         ___    " << "\n";
   cout << "
.-''-.           /    /     .'/   \\  " << "\n";
   cout << "    __  __   ___                  .           __.....__
.--.                          .' .-.  )          /    /      / /      \\ " << "\n";
   cout << "    |  |/  `.'   `.              .'|        .-''          '.
|__|                         / .'  / /          /    /       | |      | " << "\n";
   cout << "    |   .-.  .-.   '           .'  |       /      .-\'\'\"\'-.  `.
.--.     .|               (_/   / /          /    /        | |      | " << "\n";
   cout << "    |  |  |  |  |  |    __   <    |     /      /________\\   \\
|  |   .' |_                    / /          /    /  __     |/`.   .' " << "\n";
   cout << "    |  |  |  |  |  | .:--.'.  |   | ____|                  |          |
| .'     |                  / /          /    /  |  |    `.|   |  " << "\n";
   cout << "    |  |  |  |  |  |/ |   \\ | |   | \\ .\'\\    .-------------'
|  |'--.  .-'                . '          /    '   |  |     ||___|  " << "\n";
   cout << "    |  |  |  |  |  |`\" __ | | |   |/  .  \\     '-.____...---.
|  |   |  |                  / /    _.-')/    '----|  |---. |/___/  " << "\n";
   cout << "    |__|  |__|  |__| .'.''| | |    /\\  \\  `.              .'
|__|   |  |                .' '  _.'.-''/           |  |   | .'.--.  " << "\n";
   cout << "                      / /   | |_|   |  \\  \\   `''-..... -'
|  '.'            /  /.-'_.'    '----------|  |---'| |    | " << "\n";
   cout << "                      \\ \\._,\\ '/'    \\  \\  \\
|   /            /    _.'                  |  |    \\_\\    / " << "\n";
   cout <<"                      `--'  `\"\'------\'  \'---\'
`'-'            ( _.-'                    /____\\    `\'\'--\'  " << "\n\n";
   cout << "\'\'\'\'\'\'\'\'\n";
   cout << " The 24 card game is an arithmetic card game \n with the aim of finding
ways to operate 4 random card numbers \n so that the final result is 24.\n\n" ;
   cout << "\'\'\'\'\'\'\'\'\n\n";
}



int main(){
   float a, b, c, d;
   bool running = 1;
   char ops[4] = {'+', '-', '*', '/'};
   clock_t start, end;
```

```
   SPLASHSCREEN();
   while(running){
       int ressum = 0;
       int opt;
       cout << "Choose an option (1/2/3):\n";
       cout << "1. Generate random cards\n";
       cout << "2. Input my own cards\n";
       cout << "3. Exit\n";
       cin >> opt;
       switch(opt){
           case 1: {
               int ai, bi, ci, di;
               cout << "Generating random cards...\n";
               srand(time(0));
               ai = rand() % 13 + 1;
               bi = rand() % 13 + 1;
               ci = rand() % 13 + 1;
               di = rand() % 13 + 1;
               cout << "1: " << ai << "\n";
               cout << "2: " << bi << "\n";
               cout << "3: " << ci << "\n";
               cout << "4: " << di << "\n";
               a = (float) ai;
               b = (float) bi;
               c = (float) ci;
               d = (float) di;
               start = clock();
               GROUPING1(a, b, c, d, ops);
               GROUPING2(a, b, c, d, ops);
               GROUPING3(a, b, c, d, ops);
               GROUPING4(a, b, c, d, ops);
               GROUPING5(a, b, c, d, ops);
               end = clock();
               cout << "\n" << setres.size() << " solution(s) found\n";
               int i = 0;
               unordered_set<string> :: iterator itr;
               for (itr = setres.begin(); itr != setres.end(); itr++){
                   if(i % 4 == 0 || i % 4 == 1 || i % 4 == 2 || i == setres.size() -
1) cout << *itr << "  ";
                   else cout << *itr << "\n";
                   i++;
```

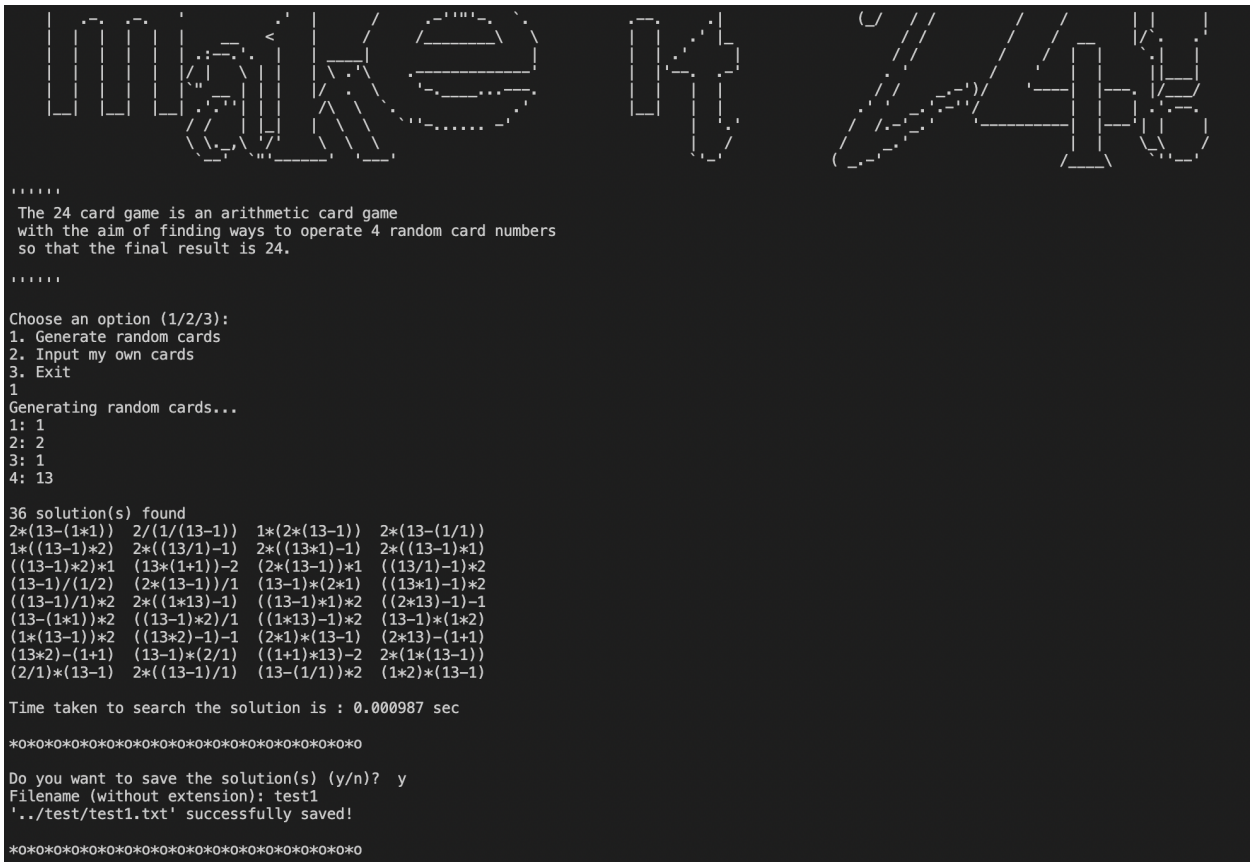
```
                }
                cout << "\n\n";
                double time_taken = double (end - start) / double(CLOCKS_PER_SEC);
                cout << "Time taken to search the solution is : " << fixed <<
time_taken << setprecision(5) << " sec\n\n";
                cout << "*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o\n\n";
                bool invalid = 1;
                invalid = 1;
                while(invalid){
                    cout << "Do you want to save the solution(s) (y/n)?  ";
                    char saveopt;
                    cin >> saveopt;
                    if(saveopt == 'y'){
                        invalid = 0;
                        cout << "Filename (without extension): ";
                        string filename;
                        cin >> filename;
                        ofstream File("../test/" + filename + ".txt");
                        unordered_set<string> :: iterator itr;
                        for (itr = setres.begin(); itr != setres.end(); itr++){
                            if(i % 4 == 0 || i % 4 == 1 || i % 4 == 2 || i ==
setres.size() - 1) File << *itr << "  ";
                            else File << *itr << "\n";
                            i++;
                        }
                        cout << "\'../test/" << filename << ".txt\' successfully
saved!\n\n";
                        cout << "*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o\n\n";
                    }
                    else if (saveopt == 'n') invalid = 0;
                    else cout << "Invalid input!\n";
                }
                break;
            }
            case 2: {
                string ai, bi, ci, di;
                bool invalid = 1;
                while(invalid){
                    cout << "Write your 4 cards!\n";
                    cout << "Write in the form of: A/1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
J, Q, K\n";
```

```
                    cout << "1: ";
                    cin >> ai;
                    cout << "2: ";
                    cin >> bi;
                    cout << "3: ";
                    cin >> ci;
                    cout << "4: ";
                    cin >> di;
                    a = INPUT_CONVERTER(ai);
                    b = INPUT_CONVERTER(bi);
                    c = INPUT_CONVERTER(ci);
                    d = INPUT_CONVERTER(di);
                    if(a == -1 || b == -1 || c == -1 || d == -1){
                        cout << "Invalid input!\n";
                        invalid = 1;
                    }
                    else invalid = 0;
                }
                start = clock();
                GROUPING1(a, b, c, d, ops);
                GROUPING2(a, b, c, d, ops);
                GROUPING3(a, b, c, d, ops);
                GROUPING4(a, b, c, d, ops);
                GROUPING5(a, b, c, d, ops);
                end = clock();
                cout << "\n" << setres.size() << " solution(s) found\n";
                int i = 0;
                unordered_set<string> :: iterator itr;
                for (itr = setres.begin(); itr != setres.end(); itr++){
                    if(i % 4 == 0 || i % 4 == 1 || i % 4 == 2 || i == setres.size() -
1) cout << *itr << "  ";
                    else cout << *itr << "\n";
                    i++;
                }
                cout << "\n\n";
                double time_taken = double (end - start) / double(CLOCKS_PER_SEC);
                cout << "Time taken to search the solution is : " << fixed <<
time_taken << setprecision(5) << " sec\n\n";
                cout << "*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o\n\n";
                invalid = 1;
                while(invalid){
```

```
                    cout << "Do you want to save the solution(s) (y/n)?  ";
                    char saveopt;
                    cin >> saveopt;
                    if(saveopt == 'y'){
                        invalid = 0;
                        cout << "Filename (without extension): ";
                        string filename;
                        cin >> filename;
                        ofstream File("../test/" + filename + ".txt");
                        unordered_set<string> :: iterator itr;
                        for (itr = setres.begin(); itr != setres.end(); itr++){
                            if(i % 4 == 0 || i % 4 == 1 || i % 4 == 2 || i ==
setres.size() - 1) File << *itr << "  ";
                            else File << *itr << "\n";
                            i++;
                        }
                        cout << "\'../test/" << filename << ".txt\' successfully
saved!\n\n";
                        cout << "*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o*o\n\n";
                    }
                    else if (saveopt == 'n') invalid = 0;
                    else cout << "Invalid input!\n";
                }
                break;
            }
            case 3: {
                cout << "exiting program...";
                running = 0;
                break;
            }
            default:{
                cout << "Invalid input!\n";
                break;
            }
        }
    }
    return 0;
}
```

Hasil Pengujian

1. Pengujian pada opsi *random genarate*



```
''''''
 The 24 card game is an arithmetic card game
 with the aim of finding ways to operate 4 random card numbers
 so that the final result is 24.

''''''

Choose an option (1/2/3):
1. Generate random cards
2. Input my own cards
3. Exit
1
Generating random cards...
1: 1
2: 2
3: 1
4: 13

36 solution(s) found
2*(13-(1*1))  2/(1/(13-1))  1*(2*(13-1))  2*(13-(1/1))
1*((13-1)*2)  2*((13/1)-1)  2*((13*1)-1)  2*((13-1)*1)
((13-1)*2)*1  (13*(1+1))-2  (2*(13-1))*1  ((13/1)-1)*2
(13-1)/(1/2)  (2*(13-1))/1  (13-1)*(2*1)  ((13*1)-1)*2
((13-1)/1)*2  2*((1*13)-1)  ((13-1)*1)*2  ((2*13)-1)-1
(13-(1*1))*2  ((13-1)*2)/1  ((1*13)-1)*2  (13-1)*(1*2)
(1*(13-1))*2  ((13*2)-1)-1  (2*1)*(13-1)  (2*13)-(1+1)
(13*2)-(1+1)  (13-1)*(2/1)  ((1+1)*13)-2  2*(1*(13-1))
(2/1)*(13-1)  2*((13-1)/1)  (13-(1/1))*2  (1*2)*(13-1)

Time taken to search the solution is : 0.000987 sec

*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O

Do you want to save the solution(s) (y/n)?  y
Filename (without extension): test1
'../test/test1.txt' successfully saved!

*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O*O
```

2. Pengujian pada opsi *random genarate*

```
Choose an option (1/2/3):
1. Generate random cards
2. Input my own cards
3. Exit
1
Generating random cards...
1: 1
2: 12
3: 5
4: 4

24 solution(s) found
4*(12-(1+5))  4*(12-(5+1))  (12-(5+1))*4  12*(1+(5-4))
12*((1-4)+5)  ((12-1)-5)*4  4*((12-5)-1)  (5+(1-4))*12
(1+(5-4))*12  12*((5+1)-4)  (5-(4-1))*12  12*(1-(4-5))
12*((5-4)+1)  12*(5+(1-4))  ((1-4)+5)*12  (12-(1+5))*4
(1-(4-5))*12  ((5+1)-4)*12  12*((1+5)-4)  ((1+5)-4)*12
((5-4)+1)*12  4*((12-1)-5)  ((12-5)-1)*4  12*(5-(4-1))

Time taken to search the solution is : 0.000624 sec

*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0

Do you want to save the solution(s) (y/n)?  y
Filename (without extension): test2
'../test/test2.txt' successfully saved!

*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0
```

3. Pengujian pada opsi *random genarate*

```
Choose an option (1/2/3):
1. Generate random cards
2. Input my own cards
3. Exit
1
Generating random cards...
1: 3
2: 12
3: 10
4: 2

76 solution(s) found
12*(10/(2+3))  10*(12/(3+2))  12*(10/(3+2))  12/((2+3)/10)
10/((3+2)/12)  3*((2*10)-12)  (12/(2+3))*10  (10/(3+2))*12
(3*(2+10))-12  ((12*3)-2)-10  ((10/2)-3)*12  ((10+2)*3)-12
((10*2)-12)*3  ((12*3)-10)-2  ((3*12)-10)-2  (12/(3+2))*10
(3*12)-(2+10)  (12/3)+(2*10)  (12*3)-(2+10)  2*(13-(1*1))
(2*10)+(12/3)  (2/1)*(13-1)  ((1+1)*13)-2  (12*10)/(2+3)
(10*12)/(3+2)  (12*10)/(3+2)  ((13/1)-1)*2  (12/3)+(10*2)
(12*3)-(10+2)  (3*10)-(12/2)  2/(1/(13-1))  (13-1)/(1/2)
1*(2*(13-1))  (10*2)+(12/3)  ((3*12)-2)-10  2*(13-(1/1))
10/((2+3)/12)  (13-(1/1))*2  1*((13-1)*2)  2*((13/1)-1)
2*((13*1)-1)  12/((3+2)/10)  2*((13-1)*1)  ((13*1)-1)*2
((13-1)*2)*1  (13*(1+1))-2  ((2+10)*3)-12  (2*(13-1))*1
(13-1)*(2/1)  (10*12)/(2+3)  2*(1*(13-1))  (2*(13-1))/1
(1*(13-1))*2  (13-1)*(2*1)  (3*(10+2))-12  ((2*10)-12)*3
((13-1)/1)*2  2*((1*13)-1)  ((13-1)*2)/1  ((13-1)*1)*2
((2*13)-1)-1  (13-(1*1))*2  12*((10/2)-3)  ((1*13)-1)*2
(10/(2+3))*12  (13-1)*(1*2)  10*(12/(2+3))  3*((10*2)-12)
(10*3)-(12/2)  ((13*2)-1)-1  (2*1)*(13-1)  (2*13)-(1+1)
(3*12)-(10+2)  (13*2)-(1+1)  2*((13-1)/1)  (1*2)*(13-1)

Time taken to search the solution is : 0.00086 sec

*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0

Do you want to save the solution(s) (y/n)?  y
Filename (without extension): test3
'../test/test3.txt' successfully saved!

*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0*0
```

4. Pengujian pada opsi *user input*

```
Choose an option (1/2/3):
1. Generate random cards
2. Input my own cards
3. Exit
2
Write your 4 cards!
Write in the form of: A/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
1: 1
2: 2
3: 3
4: 4

318 solution(s) found
4*(1*(2*3))  2*(4/(1/3))  4*(2*(1*3))  4*(2+(1+3))
3/(1/(4*2))  3*(1*(4*2))  4*(3*(1*2))  1*(3*(4*2))
4*(3+(1+2))  3*(4*(1*2))  1*(4*(3*2))  4/(1/(3*2))
1*(4*(2*3))  4*(1+(3+2))  4*(2+(3+1))  2*(4*(3/1))
2*(4*(3*1))  3*(4*(2/1))  3*(4*(2*1))  4*(3*(2/1))
4*(3*(2*1))  4*(3+(2+1))  3*(2*(4/1))  2*(3/(1/4))
2*(4*(1*3))  3*(2/(1/4))  1*(3*(2*4))  3/(1/(2*4))
3*(1*(2*4))  2/(1/(3*4))  2*(1*(3*4))  1*(2*(3*4))
4*((1*2)*3)  4*(2/(1/3))  4*((1+2)+3)  1*((4*2)*3)
2*((4*1)*3)  4*((2/1)*3)  4*((2*1)*3)  3*((1*4)*2)
1*((3*4)*2)  4*((3*1)*2)  4*((3+1)+2)  1*((4*3)*2)
4/((1/3)/2)  4*((1*3)*2)  4*((1+3)+2)  4*((2*3)*1)
2*(1*(4*3))  4*((2+3)+1)  2*((4*3)*1)  1*(2*(4*3))
3*((4*2)/1)  3*((4*2)*1)  4*((3*2)/1)  2*((3*4)/1)
3*((2*4)/1)  2*((3*1)*4)  1*((3*2)*4)  2*((1*3)*4)
1*((2*3)*4)  (4/(1/2))*3  (4*(1*2))*3  (2*(4*1))*3
(4*(2*1))*3  1*((2*4)*3)  (1*(2*4))*3  (2/(1/4))*3
(3*(1*4))*2  3*((2*4)*1)  (1*(3*4))*2  (4*(3*1))*2
(3*(4/1))*2  (4/(1/3))*2  (4*(1*3))*2  (4*(2*3))/1
(4*(2*3))*1  (2*(4*3))/1  (2*(4*3))*1  (3*(4*2))/1
(4*(3*2))/1  (2*(3*4))*1  (3*4)*(2/1)  (3+1)*(4+2)
2*(13-(1*1))  (3*2)/(1/4)  ((2*4)*1)*3  (3*(2*4))/1
(4*(2/1))*3  (3*(2/1))*4  (3*(2*1))*4  (10*2)+(12/3)
(1*(3*2))*4  (1+(3+2))*4  (3/(1/2))*4  (3+(2+1))*4
(2*(1*3))*4  (2*(3*4))/1  (1*(2*3))*4  (3/(1/4))*2
((4/1)*2)*3  (3*2)*(1*4)  ((4*1)*2)*3  (13-1)*(2/1)
((1*4)*2)*3  ((1*2)*4)*3  4*((3*2)*1)  (12*10)/(3+2)
```

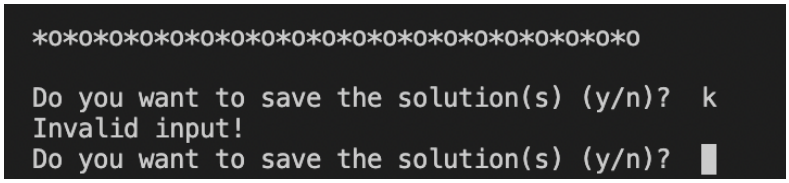
5. Pengujian pada opsi *user input*

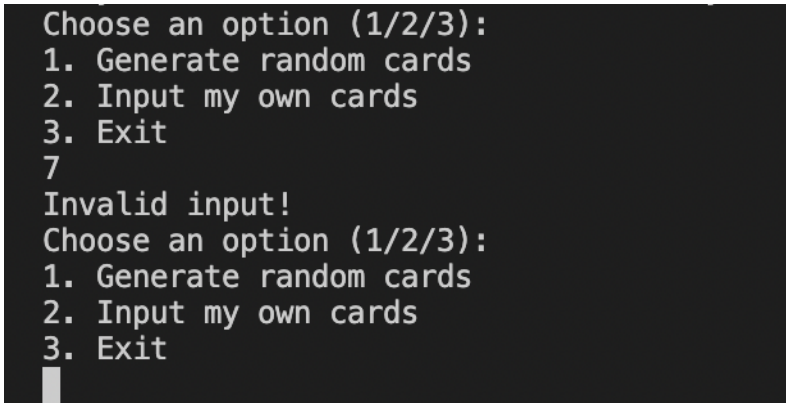
```
Choose an option (1/2/3):
1. Generate random cards
2. Input my own cards
3. Exit
2
Write your 4 cards!
Write in the form of: A/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
1: A
2: K
3: Q
4: J

350 solution(s) found
12*(13-(11*1))  12*((13/1)-11)  12*((13*1)-11)  12*((1*13)-11)
(13-(11*1))*12  (12*(13-11))*1  ((13-11)/1)*12  ((13/1)-11)*12
((13*1)-11)*12  (13-11)/(1/12)  (13-11)*(1*12)  4*(1*(2*3))
2*(4/(1/3))  4*(2*(1*3))  4*(2+(1+3))  3/(1/(4*2))
1*(12*(13-11))  3*(1*(4*2))  4*(3*(1*2))  1*(3*(4*2))
4*(3+(1+2))  3*(4*(1*2))  1*(4*(3*2))  4/(1/(3*2))
1*(4*(2*3))  4*(1+(3+2))  4*(2+(3+1))  2*(4*(3/1))
2*(4*(3*1))  3*(4*(2/1))  3*(4*(2*1))  4*(3*(2/1))
4*(3*(2*1))  4*(3+(2+1))  3*(2*(4/1))  2*(3/(1/4))
2*(4*(1*3))  3*(2/(1/4))  1*(3*(2*4))  (12/1)*(13-11)
3/(1/(2*4))  3*(1*(2*4))  2/(1/(3*4))  2*(1*(3*4))
1*(2*(3*4))  4*((1*2)*3)  4*(2/(1/3))  4*((1+2)+3)
1*((4*2)*3)  2*((4*1)*3)  4*((2/1)*3)  4*((2*1)*3)
3*((1*4)*2)  12*(1*(13-11))  ((13-11)*12)*1  1*((3*4)*2)
4*((3*1)*2)  4*((3+1)+2)  1*((4*3)*2)  (1*12)*(13-11)
4/((1/3)/2)  4*((1*3)*2)  4*((1+3)+2)  12*(13-(1*11))
4*((2*3)*1)  2*(1*(4*3))  4*((2+3)+1)  2*((4*3)*1)
1*(2*(4*3))  3*((4*2)/1)  3*((4*2)*1)  4*((3*2)/1)
```

6. Pengujian pada input jenis kartu invalid

```
Choose an option (1/2/3):
1. Generate random cards
2. Input my own cards
3. Exit
2
Write your 4 cards!
Write in the form of: A/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
1: 1
2: 3
3: 5
4: 456
Invalid input!
Write your 4 cards!
Write in the form of: A/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
1:
```

7. Pengujian pada input opsi invalid





Link to Repository

https://github.com/Putinabillaa/Tucil1_13521088