LAPORAN TUGAS KECIL 1
IF 2211

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2022/2023

Disusun oleh:

Haidar Hamda **13521105**

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG
2023

1. Algoritma Brute Force

Langkah-langkah penyelesaian permainan kartu 24 dengan algoritma brute force yang telah dibuat sebagai berikut:

1. Bentuk ekspresi matematika dengan penulisan infix.
2. Ubah ekspresi infix menjadi ekspresi postfix.
3. Hitung ekspresi postfix.
4. Simpan ekspresi jika hasil perhitungan bernilai 24.
5. Ulangi langkah 1-4 sampai semua kombinasi telah dihitung.

2. Source Program

1. main.cpp

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string>
#include <vector>
#include <limits>
#include "stdio.h"
#include <stack>
#include "string.h"
#include <fstream>
#include <iterator>
#include <chrono>
using namespace std;
bool isValid(float a,float b,float c,float d);
void f2s(float a, float b, float c, float d,std::vector<string> *p);
std::string getOP(int x);
void getKurbuk(int calc,std::string *kurbuk1,std::string *kurbuk2,std::string *kurbuk3,std::string *kurbuk4,std::string *kurbuk5);
void getKurtup(int calc,std::string *kurbuk1,std::string *kurbuk2,std::string *kurbuk3,std::string *kurbuk4,std::string *kurbuk5);
void pusVec(int x,int y,int z,int calc,float a, float b, float c, float d, std::vector<std::string> *buffer,bool tf);
void swap(float a, float b, float c, float d, float *e, float *f, float *g, float *h,int tag);
void removeDupe(std::vector<std::string> *buffer);
bool cekStr(std::string token);
void getValidInput(std::string a1, std::vector<float> *abcd);
std::vector<std::string> split(std::string str, std::string token);
void saveFile(std::string name,std::vector<std::string> buffer);
void infixtoPostfix(string exp, vector<string> *buffer);
int getPrecedence(string ops);
float getNum(string str);
float calculatePostfix(string exp);
```

```cpp
int main(){
    srand( seed: (int)time( timer: 0));
    std::vector<std::string> buffer;
    std::vector<std::string> temp;
    float a;
    float b;
    float c;
    float d;
    std::vector<float> abcd;
    std::string a1="";
    string menu="";
    std::cout << "1. input\n2. random\n";
    std::cin >> menu;
    while (menu!="1" && menu!="2"){
        std::cout<<"tidak valid!, ulangi input\n";
        std::cin >> menu;
//        std::cin.clear();
//        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }
    if (menu=="1"){
        getValidInput(a1,&abcd);
        a=abcd[0];
        b=abcd[1];
        c=abcd[2];
        d=abcd[3];
    } else if (menu=="2"){
        a=1+rand()%13;
        b=1+rand()%13;
        c=1+rand()%13;
        d=1+rand()%13;
        std::cout << "4 angka random:\n";
        std::cout << a<<" " << b<<" " << c<<" " << d<<"\n";
    }
    auto begin :time_point<...> = std::chrono::high_resolution_clock::now();
    int x=0;
```

```
    int y=0;
    int z=0;
    int calc=0;
    float atmp,btmp,ctmp,dtmp;
    atmp=a;
    btmp=b;
    ctmp=c;
    dtmp=d;
    vector<string> result;
    int tag=1;
//    int calc=0;
    while (tag<25) {
        x=0;
        while (x < 4) {
            y = 0;
            while (y < 4) {
                z = 0;
                while (z < 4) {
                    calc=0;
                    while (calc<11){
                        pusVec(x,y,z,calc, a: atmp, b: btmp, c: ctmp, d: dtmp,&buffer, tf: false);
                        infixtoPostfix( exp: buffer.at( n: buffer.size()-1), buffer: &temp);
                        if (24- calculatePostfix( exp: temp[temp.size()-1])==0){
                            pusVec(x,y,z,calc, a: atmp, b: btmp, c: ctmp, d: dtmp, buffer: &result, tf: true);
                        }
                        buffer.pop_back();
                        temp.pop_back();
                        calc++;
                    }
                    z++;
                }
                y++;
            }
            x++;
        }
```

```
101     //        printf("%f %f %f %f\n",atmp,btmp,ctmp,dtmp);
102             swap(a,b,c,d, e: &atmp, f: &btmp, g: &ctmp, h: &dtmp,tag);
103     //        printf("%f %f %f %f\n",atmp,btmp,ctmp,dtmp);
104             tag++;
105         }
106         removeDupe( buffer: &result);
107         int i=0;
108         int n=result.size();
109         printf( format: "%d solusi\n",n);
110         while (i<result.size()){
111             std::cout<<result[i]<<"\n";
112             i++;
113         }
114         auto end :time_point<...>  = std::chrono::high_resolution_clock::now();
115         auto elapsed :duration<...>  = std::chrono::duration_cast<std::chrono::nanoseconds>( d: end - begin);
116 
117         string save="";
118         std::cout << "simpan file solusi?\n"<<"1. simpan\n2. tidak\n";
119         std::cin >> save;
120         while (save!="1" && save!="2"){
121             std::cout<<"tidak valid!, ulangi input\n";
122             std::cin >> save;
123     //        std::cin.clear();
124     //        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
125         }
126         if (save=="1"){
127             std::cout<<"masukkan nama file\n";
128             std::string fileName;
129             std::cin>>fileName;
130             saveFile( name: fileName, buffer: result);
131     //        saveFile(fileName+"1",temp);
132     //        saveFile(fileName+"2",result);
133         }
134         printf( format: "waktu eksekusi: %.3f detik.\n", elapsed.count() * 1e-9);
135         return 0;
```

```
void getValidInput(std::string a1, std::vector<float> *abcd){
    std::string atmp;
    std::vector<std::string> p;
    bool isvalid= false;
    std::cout<<"masukkan 4 angka\n";
    while (!isvalid){
//        cout<<"1";
        getline( &: std::cin, &: a1);
        if (a1.empty()){
            isvalid= false;
            continue;
        }
        else{
//            cout<<"1";
            atmp=a1;
            p= split( str: atmp, token: " ");
//            for (int i = 0; i < p.size(); ++i) {
//                cout<<p[i]<<"\n";
//            }
//            cout<<p[0]<<"\n"<<p[1]<<"\n"<<p[2]<<"\n"<<p[3]<<"\n";
            for (int i = 0; i < 4; ++i) {
                if (!cekStr( token: p[i])){
                    isvalid= false;
                    break;
                } else{
                    isvalid= true;
                }
            }
        }
        if (!isvalid){
            std::cout<<"input tidak valid\n";
        }
    }
    int i=0;
    while(i<4){
```

```
    while(i<4){
        try {
            abcd->push_back((float ) stoi( str: p[i]));

        } catch(std::invalid_argument) {
            if (p[i]=="A"){
                abcd->push_back(1);
            } else if(p[i]=="J"){
                abcd->push_back(11);
            } else if(p[i]=="Q"){
                abcd->push_back(12);
            } else if(p[i]=="K"){
                abcd->push_back(13);
            }
        }
        i++;
    }
//    while(((a1->size()>1 || a1->size()<1) || (b1->size()>1 || b1->size()<1) || (c1->size()>1 || c1->size()<1) || (d1->
//        cout<<"input tidak valid";
//        cin >> *a1 >> *b1 >> *c1 >> *d1;
//    }

}
```

```cpp
void saveFile(std::string name,std::vector<std::string> buffer){
    std::string path="../test/"+name+".txt";
    std::ofstream output_file( s: path);
    std::ostream_iterator<std::string> output_iterator( &: output_file,  c: "\n");
    std::copy( first: buffer.begin(),  last: buffer.end(),  result: output_iterator);
}

std::vector<std::string> split(std::string str, std::string token){
    std::vector<std::string>result;
    while(str.size()){
        int index = str.find( str: token);
        if(index!=std::string::npos){
            result.push_back(str.substr( pos: 0,  n: index));
            str = str.substr( pos: index+token.size());
            if(str.size()==0)result.push_back(str);
        }else{
            result.push_back(str);
            str = "";
        }
    }
    return result;
}


bool cekStr(std::string token){
    return (token=="A")||(token=="2")||(token=="3")||(token=="4")||(token=="5")||(token=="6")||(token=="7")||(token=="8")||(token=="9")||(token=="10")||(token=="J")||(token=="Q")||(token=="K");
}

bool isValid(float a,float b,float c,float d){
    return (a>0&&a<14)&&(b>0&&b<14)&&(c>0&&c<14)&&(d>0&&d<14);
}
```

```cpp
void swap(float a, float b, float c, float d, float *e, float *f, float *g, float *h,int tag){
    if (tag==0){
        *e=a;
        *f=b;
        *g=c;
        *h=d;
    } else if (tag==1){
        *e=a;
        *f=d;
        *g=b;
        *h=c;
    } else if (tag==2){
        *e=a;
        *f=d;
        *g=b;
        *h=c;
    } else if (tag==3){
        *e=a;
        *f=b;
        *g=d;
        *h=c;
    } else if (tag==4){
        *e=a;
        *f=c;
        *g=b;
        *h=d;
    } else if (tag==5){
        *e=a;
        *f=c;
        *g=d;
        *h=b;
    } else if (tag==6){
        *e=a;
        *f=d;
        *g=c;
```

```
295     } else if (tag==6){
296         *e=a;
297         *f=d;
298         *g=c;
299         *h=b;
300     } else if (tag==7){
301         *e=b;
302         *f=a;
303         *g=c;
304         *h=d;
305     } else if (tag==8){
306         *e=b;
307         *f=c;
308         *g=a;
309         *h=d;
310     } else if (tag==9){
311         *e=b;
312         *f=c;
313         *g=d;
314         *h=a;
315     } else if (tag==10){
316         *e=b;
317         *f=d;
318         *g=a;
319         *h=c;
320     } else if (tag==11){
321         *e=b;
322         *f=a;
323         *g=d;
324         *h=c;
325     } else if (tag==12){
326         *e=b;
327         *f=d;
328         *g=c;
329         *h=a;
```

```
    } else if (tag==13){
        *e=c;
        *f=b;
        *g=d;
        *h=a;
    } else if (tag==14){
        *e=c;
        *f=d;
        *g=b;
        *h=a;
    } else if (tag==15){
        *e=c;
        *f=d;
        *g=a;
        *h=b;
    } else if (tag==16){
        *e=c;
        *f=b;
        *g=a;
        *h=d;
    } else if (tag==17){
        *e=c;
        *f=a;
        *g=b;
        *h=d;
    } else if (tag==18){
        *e=c;
        *f=a;
        *g=d;
        *h=b;
    } else if (tag==19){
        *e=d;
        *f=c;
        *g=a;
        *h=b;
    } else if (tag==20){
        *e=d;
        *f=a;
        *g=c;
        *h=b;
    } else if (tag==21){
        *e=d;
        *f=a;
        *g=b;
        *h=c;
    } else if (tag==22){
        *e=d;
        *f=c;
        *g=b;
        *h=a;
    } else if (tag==23){
        *e=d;
        *f=b;
        *g=c;
        *h=a;
    } else if (tag==24) {
        *e = d;
        *f = b;
        *g = a;
        *h = c;
    }
}
```

```cpp
void pusVec(int x,int y,int z,int calc,float a, float b, float c, float d, std::vector<std::string> *buffer,bool tf){
    std::vector<std::string> p;
    f2s(a,b,c,d,&p);
    std::string op1,op2,op3;
    op1=getOP(x);
    op2=getOP( x: y);
    op3=getOP( x: z);
    std::string kurbuk1,kurbuk2,kurbuk3,kurbuk4,kurbuk5;
    getKurbuk(calc,&kurbuk1,&kurbuk2,&kurbuk3,&kurbuk4,&kurbuk5);
    std::string kurtup1,kurtup2,kurtup3,kurtup4,kurtup5;
    getKurtup(calc, kurbuk1: &kurtup1, kurbuk2: &kurtup2, kurbuk3: &kurtup3, kurbuk4: &kurtup4, kurbuk5: &kurtup5);
//    string cc= to_string(calc);
//    string xx= to_string(x);
//    string yy= to_string(y);
//    string zz= to_string(z);
//    cc+". "+xx+" "+yy+" "+zz+" "+
    if (tf){
        int i=0;
        while (i<4){
            if (p[i]=="S"){
                p[i]="10";
            }
            i++;
        }
    }
    std::string buf=kurbuk1+kurbuk2+p[0]+op1+kurbuk3+kurbuk4+ p[1]+kurtup1+op2+kurbuk5+p[2]+kurtup2+kurtup3+op3+p[3]+kurtup4+kurtup5;
    std::string s(buf);


//    for (int i = 0; i < s.size(); i++) {
//        cout << i <<". " << s[i] << "\n";
//    }
    buffer->push_back(s);
}
```

```cpp
std::string getOP(int x){
    if (x==0){
        return "+";
    } else if (x==1){
        return "-";
    } else if (x==2){
        return "*";
    } else if (x==3){
        return "/";
    } else{
        return "0";
    }
}
```

```
void getKurbuk(int calc,std::string *kurbuk1,std::string *kurbuk2,std::string *kurbuk3,std::string *kurbuk4,std::string *kurbuk5){
    *kurbuk1="";
    *kurbuk2="";
    *kurbuk3="";
    *kurbuk4="";
    *kurbuk5="";
    if (calc==0){ //a b c d
    } else if (calc==1){ //(a b) c d
        *kurbuk1="(";
    } else if (calc==2){ //(a b) (c d)
        *kurbuk1="(";
        *kurbuk5="(";
    } else if (calc==3){ //((a b) c) d
        *kurbuk1="(";
        *kurbuk2="(";
    } else if (calc==4){ // a b (c d)
        *kurbuk5="(";
    } else if (calc==5){ // a (b c) d
        *kurbuk3="(";
    } else if (calc==6){ // (a (b c)) d
        *kurbuk1="(";
        *kurbuk3="(";
    } else if (calc==7){ // a (b (c d))
        *kurbuk3="(";
        *kurbuk5="(";
    } else if (calc==8){ // a ((b c) d)
        *kurbuk3="(";
        *kurbuk4="(";
    } else if (calc==9){ // a (b c d)
        *kurbuk3="(";
    } else if (calc==10){ // (a b c) d
        *kurbuk1="(";
    }
}
```

```
void getKurtup(int calc,std::string *kurtup1,std::string *kurtup2,std::string *kurtup3,std::string *kurtup4,std::string *kurtup5){
    *kurtup1="";
    *kurtup2="";
    *kurtup3="";
    *kurtup4="";
    *kurtup5="";
    if (calc==0){ //a b c d
    } else if (calc==1){ //(a b) c d
        *kurtup1=")";
    } else if (calc==2){ //(a b) (c d)
        *kurtup1=")";
        *kurtup5=")";
    } else if (calc==3){ //((a b) c) d
        *kurtup1=")";
        *kurtup3=")";
    } else if (calc==4){ // a b (c d)
        *kurtup4=")";
    } else if (calc==5){ // a (b c) d
        *kurtup3=")";
    } else if (calc==6){ // (a (b c)) d
        *kurtup2=")";
        *kurtup3=")";
    } else if (calc==7){ // a (b (c d))
        *kurtup4=")";
        *kurtup5=")";
    } else if (calc==8){ // a ((b c) d)
        *kurtup3=")";
        *kurtup5=")";
    } else if (calc==9){ // a (b c d)
        *kurtup4=")";
    } else if (calc==10){ // (a b c) d
        *kurtup3=")";
    }
}
```

```cpp
void infixtoPostfix(string exp, vector<string> *buffer){
    vector<string> operatorr;
    string ret="";
    string s="";
    int i=0;
    while(i<exp.size()){
        s="";
        s+=exp.at( n: i);
L1:
        if (s=="+"|| s=="-" || s=="*" || s=="/" || s=="(" || s==")"){
            if (operatorr.empty()){
                operatorr.push_back(s);
            } else{

                if (s=="("){
                    operatorr.push_back(s);
                } else if (s==")"){
                    while (!operatorr.empty() && operatorr.at( n: operatorr.size()-1)!="("){
                        ret+=operatorr.at( n: operatorr.size()-1);
                        operatorr.pop_back();
                    }
//                  operatorr.pop_back();
                    if (!operatorr.empty()&&operatorr.at( n: operatorr.size()-1)=="("){
                        operatorr.pop_back();
                    }
                }
                else if (getPrecedence( ops: operatorr.at( n: operatorr.size()-1))< getPrecedence( ops: s)){
                    operatorr.push_back(s);
                } else if (getPrecedence( ops: operatorr.at( n: operatorr.size()-1))== getPrecedence( ops: s)){
                    ret+=operatorr.at( n: operatorr.size()-1);
                    operatorr.pop_back();
                    operatorr.push_back(s);
                } else if (getPrecedence( ops: operatorr.at( n: operatorr.size()-1))> getPrecedence( ops: s)){
//                    while (!operatorr.empty()&& operatorr.at(operatorr.size()-1)!="("&& getPrecedence(operatorr.at(operatorr.size()-1))> getPrecedence(s)){
//                        ret+=operatorr.at(operatorr.size()-1);
//                        operatorr.pop_back();
//                    }
//                    if (!operatorr.empty()&&operatorr.at(operatorr.size()-1)=="("){
//                        operatorr.pop_back();
//                    }
                    ret+=operatorr.at( n: operatorr.size()-1);
                    operatorr.pop_back();
                    goto L1;
                }
            }
        } else {
            if (s!="("&&s!=")"){
                ret+=s;
            }
        }
        i++;
    }
    while (!operatorr.empty()){
        ret+=operatorr.at( n: operatorr.size()-1);
        operatorr.pop_back();
    }
    buffer->push_back(ret);
}
```

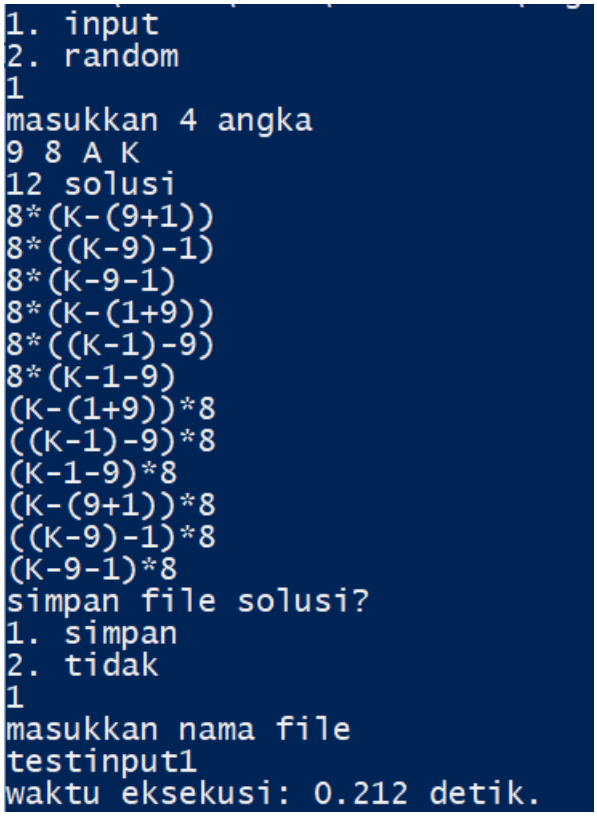
```cpp
int getPrecedence(string ops){
    if (ops=="+" || ops=="-"){
        return 1;
    } else if (ops=="*" || ops=="/"){
        return 2;
    } else {
        return 0;
    }
}

float getNum(string str){
    float f=0;
    try {
        f= stof(str);
    } catch(std::invalid_argument) {
        if (str=="S"){
            f=10;
        } else if(str=="J"){
            f=11;
        } else if(str=="Q"){
            f=12;
        } else if(str=="K"){
            f=13;
        }
    }
    return f;
}
```

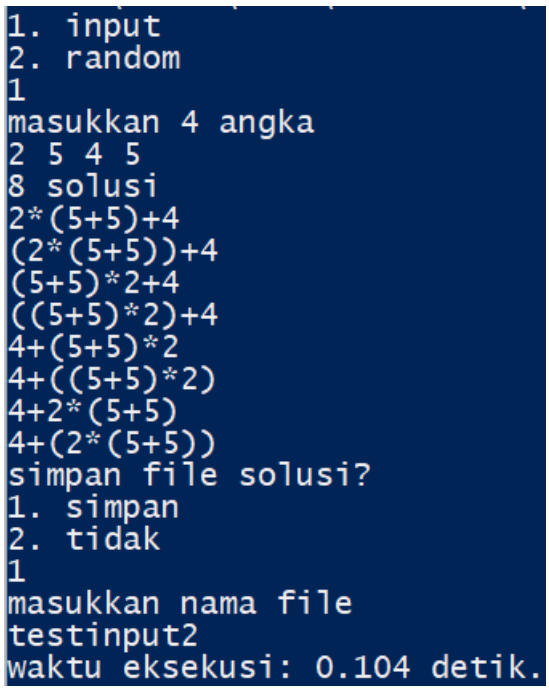
```cpp
float calculatePostfix(string exp){
    vector<float> res;
    string s="";
    int i=0;
    float o1=0;
    float o2=0;
    float result=0;
    while (i<exp.size()){
        s="";
        s+=exp[i];
        if (s=="+"||s=="-"||s=="*"||s=="/"){
            o2=res[res.size()-1];
            res.pop_back();
            o1=res[res.size()-1];
            res.pop_back();
            if (s=="+"){
                result=o1+o2;
                res.push_back(result);
            } else if (s=="-"){
                result=o1-o2;
                res.push_back(result);
            } else if (s=="*"){
                result=o1*o2;
                res.push_back(result);
            } else if (s=="/"){
                result=o1/o2;
                res.push_back(result);
            }
        } else{
            res.push_back(getNum( str: s));
        }
        i++;
    }
//    cout<<res.size()<<"\n";
    return res[0];
```

3. Input/Output

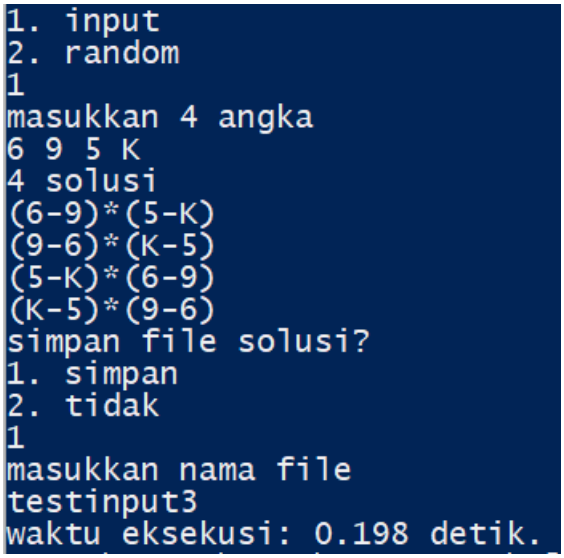
1. masukan berdasarkan input (1)



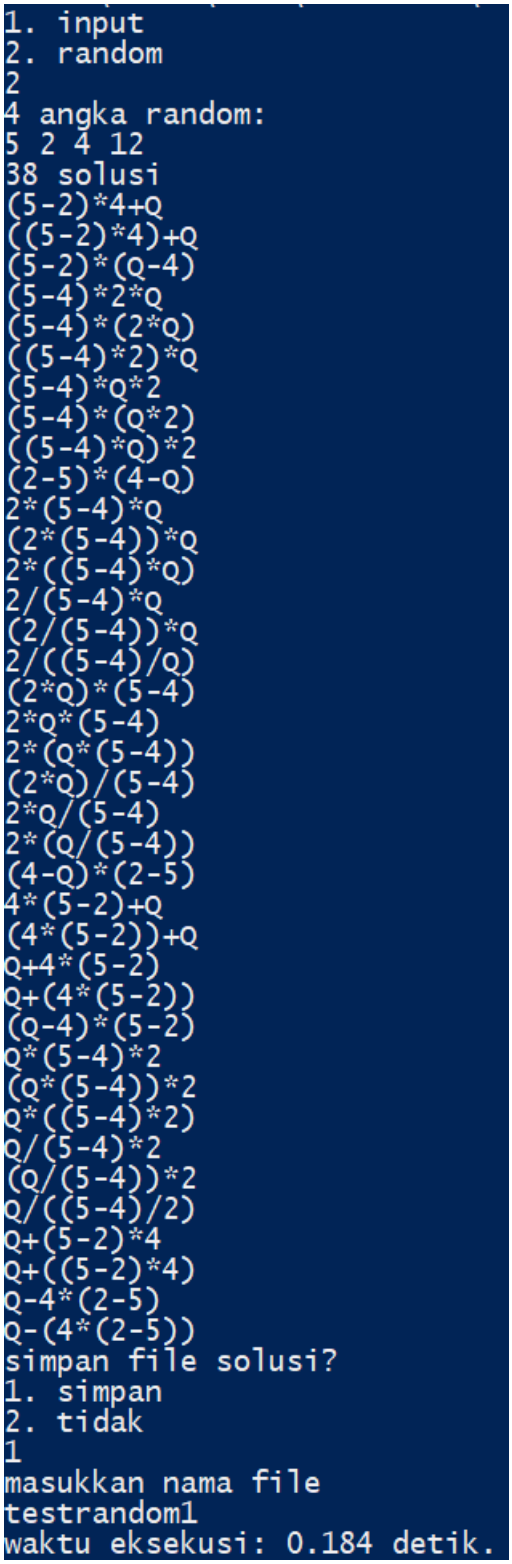
2. masukan berdasarkan input (2)



3. masukan berdasarkan input (3)



4. masukan random (1)



5. masukan random (2)

```
1. input
2. random
2
4 angka random:
10 7 1 5
26 solusi
10*((7/5)+1)
10*(7/5+1)
10*(1+(7/5))
10*(1+7/5)
(7*5)-(10+1)
7*5-(10+1)
7*5-10-1
(7*5)-10-1
((7*5)-10)-1
(7*5-10)-1
(7*5)-(1+10)
7*5-(1+10)
7*5-1-10
(7*5)-1-10
((7*5)-1)-10
(7*5-1)-10
((7/5)+1)*10
(7/5+1)*10
(1+(7/5))*10
(1+7/5)*10
(5*7)-(1+10)
5*7-(1+10)
5*7-1-10
(5*7)-1-10
((5*7)-1)-10
(5*7-1)-10
simpan file solusi?
1. simpan
2. tidak
1
masukkan nama file
testrandom2
waktu eksekusi: 0.221 detik.
```

6. masukan random (3)

```
1. input
2. random
2
4 angka random:
5 9 12 12
32 solusi
((5+9)-Q)*Q
(5+(9-Q))*Q
(5+9-Q)*Q
((5-Q)+9)*Q
(5-Q+9)*Q
(5-(Q-9))*Q
((9+5)-Q)*Q
(9+(5-Q))*Q
(9+5-Q)*Q
((9-Q)+5)*Q
(9-Q+5)*Q
(9-(Q-5))*Q
((9*Q)+Q)/5
(9*Q+Q)/5
(Q+(9*Q))/5
(Q+9*Q)/5
((Q*9)+Q)/5
(Q*9+Q)/5
Q*((9-Q)+5)
Q*(9-Q+5)
Q*(9-(Q-5))
(Q+(Q*9))/5
(Q+Q*9)/5
Q*(9+(5-Q))
Q*((9+5)-Q)
Q*(9+5-Q)
Q*(5+(9-Q))
Q*((5+9)-Q)
Q*(5+9-Q)
Q*((5-Q)+9)
Q*(5-Q+9)
Q*(5-(Q-9))
simpan file solusi?
1. simpan
2. tidak
1
masukkan nama file
testrandom3
waktu eksekusi: 0.255 detik.
```

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	✓	