TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2022/2023

PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA BRUTE FORCE

Disusun Oleh:

Arleen Chrysantha Gunardi

NIM. 13521059

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

I. Latar Belakang Persoalan

Permainan Kartu 24 merupakan salah satu permainan yang memanfaatkan nilai-nilai angka pada kartu. Permainan ini dilakukan dengan cara melakukan operasi aritmetika (pertambahan, pengurangan, perkalian, dan pembagian) terhadap empat buah angka kartu yang diambil secara acak dari dek. Operasi aritmetika tersebut dilakukan sedemikian rupa sehingga hasilnya adalah 24. Kartu A direpresentasikan sebagai angka satu (1), kartu J sebagai angka sebelas (11), kartu Q sebagai angka dua belas (12), kartu K sebagai angka tiga belas (13), dan kartu angka sesuai dengan nilai angkanya. Setiap kartu hanya boleh dioperasikan satu kali (tidak boleh dioperasikan secara berulang). Misalkan empat kartu dengan nilai 6, 7, J, dan 8 diambil secara acak dari dek, berarti beberapa solusi untuk permainan adalah ((7+11)*8)/6 atau (8/6)*(7+11).

Maka dari itu, untuk membantu menyelesaikan permainan ini, dibuatlah sebuah program "24 Game Solver" yang menampilkan semua kemungkinan solusi operasi aritmetika dari empat buah angka yang dapat menghasilkan 24. Program ini dibuat dengan memanfaatkan bahasa pemrograman C++ dan algoritma Brute Force. Secara umum, program akan meminta empat angka dari pengguna atau memberikan empat angka acak lalu menampilkan seluruh kemungkinan solusi permainan.

II. Ide Solusi Persoalan

Dengan algoritma Brute Force, tepatnya Exhaustive Search, program akan mencoba semua kemungkinan operasi untuk empat angka, empat operator, serta operasi kurung. Bentuk solusi secara umum (tanpa operasi kurung) adalah sebagai berikut.

<angka1> <operator1> <angka2> <operator2> <angka3> <operator3> <angka4>

Keempat angka tersebut dapat saling bertukar posisi, misalnya urutan angka di atas dapat ditukar menjadi angka2, angka4, angka1, angka3. Dengan aturan permutasi, dapat diketahui bahwa banyaknya kemungkinan pertukaran angka adalah sebanyak 4! = 4 * 3 * 2 * 1 = 24 kemungkinan.

Adapun langkah untuk menemukan semua kemungkinan urutan keempat angka adalah sebagai berikut. Misalkan A = angka1, B = angka2, C = angka3, dan D = angka4, sehingga urutan awal kartu adalah ABCD.

1. Tukar digit pertama dengan digit pertama. (urutan menjadi ABCD)
    a. Tukar digit kedua dengan digit kedua. (urutan menjadi ABCD)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **ABCD**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **ABDC**)
    b. Tukar digit kedua dengan digit ketiga. (urutan menjadi ACBD)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **ACBD**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **ACDB**)
    c. Tukar digit kedua dengan digit keempat. (urutan menjadi ADCB)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **ADCB**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **ADBC**)
2. Tukar digit pertama dengan digit kedua. (urutan menjadi BACD)
    a. Tukar digit kedua dengan digit kedua. (urutan menjadi BACD)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **BACD**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **BADC**)
    b. Tukar digit kedua dengan digit ketiga. (urutan menjadi BCAD)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **BCAD**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **BCDA**)
    c. Tukar digit kedua dengan digit keempat. (urutan menjadi BDCA)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **BDCA**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **BDAC**)
3. Tukar digit pertama dengan digit ketiga. (urutan menjadi CBAD)
    a. Tukar digit kedua dengan digit kedua. (urutan menjadi CBAD)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **CBAD**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **CBDA**)
    b. Tukar digit kedua dengan digit ketiga. (urutan menjadi CABD)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **CABD**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **CADB**)
    c. Tukar digit kedua dengan digit keempat. (urutan menjadi CDAB)

i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **CABD**)
ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **CADB**)

4. Tukar digit pertama dengan digit keempat. (urutan menjadi DBCA)
    a. Tukar digit kedua dengan digit kedua. (urutan menjadi DBCA)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **DBCA**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **DBAC**)
    b. Tukar digit kedua dengan digit ketiga. (urutan menjadi DCBA)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **DCBA**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **DCAB**)
    c. Tukar digit kedua dengan digit keempat. (urutan menjadi DBAC)
        i. Tukar digit ketiga dengan digit ketiga. (urutan menjadi **DBAC**)
        ii. Tukar digit ketiga dengan digit keempat. (urutan menjadi **DBCA**)

Berikut adalah tabel yang menampilkan seluruh kemungkinan urutan angka kartu.

ABCD	ABDC	ACBD	ACDB	ADCB	ADBC
BACD	BADC	BCAD	BCDA	BDCA	BDAC
CBAD	CBDA	CABD	CADB	CDAB	CDBA
DBCA	DBAC	DCBA	DCAB	DBAC	DBCA

Tabel 2.1 Urutan Angka Kartu

Selain itu, operator yang digunakan dalam permainan ini adalah operator tambah (+), kurang (-), kali (*), dan bagi (/). Untuk empat angka, operator yang dibutuhkan sebanyak tiga operator yang boleh berulang, sehingga banyaknya kemungkinan untuk urutan operator adalah sebanyak 4 * 4 * 4 = 64 kemungkinan.

Langkah untuk menemukan seluruh kemungkinan urutan operator serupa dengan langkah untuk menemukan seluruh kemungkinan urutan angka kartu, namun hanya tiga “digit”. Berikut adalah tabel yang menampilkan seluruh kemungkinan urutan operator.

+++	++-	++*	++/	+-+	+--	+-*	+-/	+*+	+*-	+**	+*/	+/+	+/-	+/*	+//
-++	-+-	-+*	-+/	--+	---	--*	--/	-*+	-*-	-**	-*/	-/+	-/-	-/*	-//
*++	*+-	*+*	*+/	*-+	*--	*-*	*-/	**+	**-	***	**/	*/+	*/-	*/*	*//
/++	/+-	/+*	/+/	/-+	/--	/-*	/-/	/*+	/*-	/**	/*/	//+	//-	//*	///

Tabel 2.2 Urutan Operator

Selain angka dan operator, terdapat operasi kurung yang urutannya dapat divariasikan. Terdapat sebanyak lima (5) kemungkinan urutan operasi kurung. Berikut adalah seluruh kemungkinan urutan operasi kurung.

((A _ B) _ C) _ D

(A _ (B _ C)) _ D

(A _ B) _ (C _ D)

A _ ((B _ C) _ D)

A _ (B _ (C _ D))

Jadi, dengan 24 kemungkinan urutan kartu, 64 kemungkinan urutan operator (+, -, *, /), dan 5 kemungkinan urutan operator kurung, maka dapat diketahui bahwa program harus mengecek paling banyak 24 * 64 * 5 = 7.680 kemungkinan. Dari seluruh kemungkinan tersebut, program hanya akan menampilkan operasi-operasi yang menghasilkan nilai 24.

III. Implementasi Program

Program ditulis dalam bahasa pemrograman C++. Adapun sistematika file untuk membuat program ini terdiri dari beberapa folder sebagai berikut.

```
Tucil1_13521059
├── bin
|   └── kartu24.out
├── doc
|   └── Tucil1_13521059.pdf
├── src
|   └── kartu24.cpp
├── test
|   ├── t1.txt
|   ├── t2.txt
|   ├── t3.txt
|   ├── t4.txt
|   ├── t5.txt
|   └── t6.txt
└── README.md
```

Program utama ditulis pada file kartu24.cpp pada folder src. File executable untuk menjalankan program disimpan pada folder bin, sehingga pengguna hanya perlu menjalankan perintah "./kartu24.out" pada terminal yang berada pada directory folder bin. Hasil dari solusi permainan disimpan ke dalam file teks pada folder test. Terakhir, folder doc berisi laporan ini.

Berikut adalah implementasi dari ide solusi persoalan dalam program berupa kode program.

3.1 Inisialisasi Program

Program diawali dengan meng-*include* beberapa *library* penunjang yang dibutuhkan dalam pembuatan program. Adapun *library* yang dibutuhkan adalah tipe data string, vector, library cstdlib, chrono (untuk waktu), iostream, fstream, dan sstream.

Selain itu, terdapat pula beberapa variabel global yang digunakan pada seluruh program, yaitu list_operation dan ln_operation. Vector list_operation adalah list berisi semua operasi yang menghasilkan 24. Banyaknya elemen list_operation disimpan pada ln_operation.

```cpp
#include <string>
#include <vector>
#include <cstdlib>
#include <chrono>
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;

/* KAMUS GLOBAL */
```

```
vector<string> list_operation; //array of strings: list dari semua operasi yang menghasilkan nilai 24
int ln_operation; //jumlah solusi yang ditemukan (panjang list operation) ln_operation = list_operation.size()
```

3.2 Fungsi dan Prosedur Penunjang

3.2.1 Fungsi charToFloat

Fungsi charToFloat berfungsi untuk mengubah setiap karakter angka kartu menjadi nilai yang setara.

```
float charToFloat (string s){
    /*
    Mengubah string menjadi angka yang sepadan dengan nilai kartu
    Misalkan 'A'=1, '2'=2, 'J'=11, 'Q'=12, 'K'=13
     Jika string s bukan nilai kartu (misalnya '1', '0', atau huruf selain J,Q,K), maka mengembalikan nilai 0
    */
    if (s=="A"){
        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 0;
    }
}
```

3.2.2 Fungsi intToChar

Fungsi intToChar berfungsi untuk mengubah nilai menjadi angka kartu yang setara.

```
string intToChar (int i){
    /*
    Mengubah angka (int) menjadi string yang sepadan dengan nilai kartu
    Misalkan 'A'=1, '2'=2, 'J'=11, 'Q'=12, 'K'=13
    */
    if (i==1){
        return "A";
    } else if (i==2){
        return "2";
    } else if (i==3){
        return "3";
    } else if (i==4){
        return "4";
    } else if (i==5){
        return "5";
    } else if (i==6){
        return "6";
    } else if (i==7){
        return "7";
    } else if (i==8){
        return "8";
    } else if (i==9){
        return "9";
    } else if (i==10){
        return "10";
    } else if (i==11){
        return "J";
    } else if (i==12){
        return "Q";
    } else if (i==13){
        return "K";
    }
}
```

3.2.3 Fungsi charToOp

Fungsi charToOp berfungsi untuk menghitung hasil operasi antara operang1 dan operand2 dengan operator op.

```
float charToOp (char op, float operand1, float operand2){
    /* Menghitung operasi antara operand1 dan operand2 dengan operator op */
    if (op=='+'){
        return operand1 + operand2;
    } else if (op=='-'){
        return operand1 - operand2;
    } else if (op=='*'){
        return operand1 * operand2;
```

```
    } else if (op=='/'){
        return operand1 / operand2;
    }
}
```

3.2.4 Fungsi findElmt

Fungsi findElmt berfungsi untuk mencari apakah terdapat elmt pada vector vec.

```
bool findElmt (vector<string> vec, string elmt){
    /* Mengembalikan true jika elmt terdapat pada vector vec */
    bool found = false;
    int count = 0;
    if (vec.size()>0){
        while (!found && count <= vec.size()-1){
            if (vec[count]==elmt){
                found = true;
            } else {
                count++;
            }
        }
    }
    return found;
}
```

3.2.5 Fungsi splitString

Fungsi splitString berfungsi untuk memisahkan string setiap spasi. Fungsi ini digunakan untuk melakukan validasi input.

```
vector<string> splitString (const string &strInput){
    /* Split String dengan delimiter spasi ' ' */
    vector<string> strResult;
    char delimiter = ' ';
    stringstream ss (strInput);
    string item;
    while (getline(ss,item,delimiter)){
        strResult.push_back(item);
    }
    return strResult;
}
```

3.2.6 Prosedur brute_force

Prosedur brute_force merupakan prosedur untuk mencari semua kemungkinan solusi permainan. Prosedur ini menerima input parameter angkaKartu berupa vector of float. Pertama, akan dilakukan percobaan untuk seluruh kemungkinan urutan angka kartu berdasarkan algoritma yang telah dijelaskan pada bagian sebelumnya. Setelah

itu, dilakukan percobaan untuk seluruh kemungkinan urutan operator tambah, kurang, kali, dan bagi. Terakhir, dilakukan juga percobaan untuk seluruh kemungkinan operasi kurung. Semua operasi yang menghasilkan nilai 24 akan ditambahkan ke list_operation, yaitu vector of strings yang menyimpan semua operasi solusi. Namun, jika terdapat angka kembar dan operasi solusi tersebut kembar pula, maka operasi hanya ditambahkan ke list_operation satu kali saja. Ini berarti program hanya akan menambahkan operasi ke vector list_operation jika dan hanya jika hasilnya 24 dan belum terdapat pada list_operation.

```
void brute_force(vector<float> angkaKartu){
  /* KAMUS LOKAL */
  int counter1,counter2,counter3; //indeks permutasi urutan kartu
  float temp; //variabel untuk swap kartu
  char list_operator[4] = {'+','-','*','/'};
  int countOp1,countOp2,countOp3; //indeks permutasi urutan operator
  float hasil1,hasil2,hasil; //hasil perhitungan
  /* ALGORITMA */
  float angka0 = angkaKartu[0]; //fixed urutan angka asli
  float angka1 = angkaKartu[1];
  float angka2 = angkaKartu[2];
  float angka3 = angkaKartu[3];
  /* Permutasi 4 Kartu */
  for (counter1=0; counter1<=3; counter1++){
    angkaKartu[0] = angka0; //kembalikan ke urutan kartu aslinya
    angkaKartu[1] = angka1;
    angkaKartu[2] = angka2;
    angkaKartu[3] = angka3;
    /* Swap angkaKartu[0] dengan elemen indeks 0,1,2,3 */
    temp = angkaKartu[0];
    angkaKartu[0] = angkaKartu[counter1];
    angkaKartu[counter1] = temp;

    float angkaKomb0 = angkaKartu[0]; //fixed urutan angka setelah swap pertama
    float angkaKomb1 = angkaKartu[1];
    float angkaKomb2 = angkaKartu[2];
    float angkaKomb3 = angkaKartu[3];

    for (counter2=1; counter2<=3; counter2++){
      /* Kembalikan ke urutan kartu swap pertama */
      angkaKartu[0] = angkaKomb0;
      angkaKartu[1] = angkaKomb1;
      angkaKartu[2] = angkaKomb2;
      angkaKartu[3] = angkaKomb3;
      /* Swap angkaKartu[1] dengan elemen indeks 1,2,3 */
      temp = angkaKartu[1];
      angkaKartu[1] = angkaKartu[counter2];
      angkaKartu[counter2] = temp;
```

```
      for (counter3=2; counter3<=3; counter3++){
        /* Swap angkaKartu[2] dengan elemen indeks 2,3 */
        temp = angkaKartu[2];
        angkaKartu[2] = angkaKartu[counter3];
        angkaKartu[counter3] = temp;
        /* Operasi */
        for (countOp1=0; countOp1<=3; countOp1++){
          for (countOp2=0; countOp2<=3; countOp2++){
            for (countOp3=0; countOp3<=3; countOp3++){
              /* list_operator[countOp1] list_operator[countOp2] list_operator[countOp3] */
              /* jenis operasi kurung:
              ((A  _  B) _  C) _ D
              ( A  _ (B  _  C))_ D
               (A  _  B) _ (C  _ D)
                A  _((B  _  C) _ D)
                A  _( B  _ (C  _ D))
              */
              /* ((A  _  B) _  C) _ D */
              hasil1 = charToOp(list_operator[countOp1], angkaKartu[0], angkaKartu[1]); //(A _
B)
              hasil2 = charToOp(list_operator[countOp2], hasil1, angkaKartu[2]); //((A _ B) _
C)
              hasil = charToOp(list_operator[countOp3], hasil2, angkaKartu[3]); //((A _ B) _
C)_D
              if (hasil==24.0 && !findElmt(list_operation, "((" + std::to_string((int)
              angkaKartu[0]) + list_operator[countOp1] + std::to_string((int) angkaKartu[1]) +
              ")" + list_operator[countOp2] + std::to_string((int) angkaKartu[2]) + ")" +
              list_operator[countOp3] + std::to_string((int) angkaKartu[3]))){
                 list_operation.push_back("((" + std::to_string((int) angkaKartu[0]) +
                list_operator[countOp1] + std::to_string((int) angkaKartu[1]) + ")" +
                list_operator[countOp2] + std::to_string((int) angkaKartu[2]) + ")" +
                list_operator[countOp3] + std::to_string((int) angkaKartu[3]));
              }

              /* ( A  _ (B  _  C))_ D */
              hasil1 = charToOp(list_operator[countOp2], angkaKartu[1], angkaKartu[2]); //(B _
C)
              hasil2 = charToOp(list_operator[countOp1], angkaKartu[0], hasil1); //(A _ (B _
C))
              hasil = charToOp(list_operator[countOp3], hasil2, angkaKartu[3]); //(A _ (B _
C))_D
              if (hasil==24.0 && !findElmt(list_operation, "(" + std::to_string((int)
              angkaKartu[0]) + list_operator[countOp1] + "(" + std::to_string((int)
              angkaKartu[1]) + list_operator[countOp2] + std::to_string((int) angkaKartu[2]) +
              "))" + list_operator[countOp3] + std::to_string((int) angkaKartu[3]))){
                 list_operation.push_back("(" + std::to_string((int) angkaKartu[0]) +
                list_operator[countOp1] + "(" + std::to_string((int) angkaKartu[1]) +
                list_operator[countOp2] + std::to_string((int) angkaKartu[2]) + "))" +
                list_operator[countOp3] + std::to_string((int) angkaKartu[3]));
              }

              /* (A  _  B) _ (C  _ D) */
```

```
                hasil1 = charToOp(list_operator[countOp1], angkaKartu[0], angkaKartu[1]); //(A _
B)
                hasil2 = charToOp(list_operator[countOp3], angkaKartu[2], angkaKartu[3]); //(C _
D)
                hasil = charToOp(list_operator[countOp2], hasil1, hasil2); //(A _ B) _ (C _ D)
                if (hasil==24.0 && !findElmt(list_operation, "(" + std::to_string((int)
                angkaKartu[0]) + list_operator[countOp1] + std::to_string((int) angkaKartu[1]) +
                ")" + list_operator[countOp2] + "(" + std::to_string((int) angkaKartu[2]) +
                list_operator[countOp3] + std::to_string((int) angkaKartu[3]) + ")")){
                   list_operation.push_back("(" + std::to_string((int) angkaKartu[0]) +
                  list_operator[countOp1] + std::to_string((int) angkaKartu[1]) + ")" +
                  list_operator[countOp2] + "(" + std::to_string((int) angkaKartu[2]) +
                  list_operator[countOp3] + std::to_string((int) angkaKartu[3]) + ")");
                }

                /* A  _((B  _  C) _ D) */
                hasil1 = charToOp(list_operator[countOp2], angkaKartu[1], angkaKartu[2]); //(B _
C)
                hasil2 = charToOp(list_operator[countOp3], hasil1, angkaKartu[3]); //((B _ C) _
D)
                hasil = charToOp(list_operator[countOp1], angkaKartu[0], hasil2); //A_((B _ C) _
D)
                if (hasil==24.0 && !findElmt(list_operation, std::to_string((int) angkaKartu[0])
                + list_operator[countOp1] + "((" + std::to_string((int) angkaKartu[1]) +
                list_operator[countOp2] + std::to_string((int) angkaKartu[2]) + ")" +
                list_operator[countOp3] + std::to_string((int) angkaKartu[3]) + ")")){
                   list_operation.push_back(std::to_string((int) angkaKartu[0]) +
                  list_operator[countOp1] + "((" + std::to_string((int) angkaKartu[1]) +
                  list_operator[countOp2] + std::to_string((int) angkaKartu[2]) + ")" +
                  list_operator[countOp3] + std::to_string((int) angkaKartu[3]) + ")");
                }

                /* A  _( B  _ (C  _ D)) */
                hasil1 = charToOp(list_operator[countOp3], angkaKartu[2], angkaKartu[3]); //(C _
D)
                hasil2 = charToOp(list_operator[countOp2], angkaKartu[1], hasil1); //(B _ (C _
D))
                hasil = charToOp(list_operator[countOp1], angkaKartu[0], hasil2); //A_(B _ (C _
D))
                if (hasil==24.0 && !findElmt(list_operation, std::to_string((int) angkaKartu[0])
                + list_operator[countOp1] + "(" + std::to_string((int) angkaKartu[1]) +
                list_operator[countOp2] + "(" + std::to_string((int) angkaKartu[2]) +
                list_operator[countOp3] + std::to_string((int) angkaKartu[3]) + "))")){
                   list_operation.push_back(std::to_string((int) angkaKartu[0]) +
                  list_operator[countOp1] + "(" + std::to_string((int) angkaKartu[1]) +
                  list_operator[countOp2] + "(" + std::to_string((int) angkaKartu[2]) +
                  list_operator[countOp3] + std::to_string((int) angkaKartu[3]) + "))");
                }
              }
            }
          }
        }
```

```
    }
  }
}
```

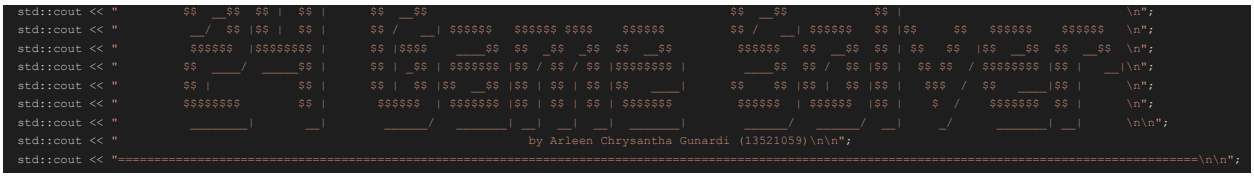
3.3 Program Utama

Program utama diawali dengan deklarasi variabel penunjang sebagai berikut.

```
int main(){
    /* KAMUS */
    string inputKartu;
    vector<string> kartu; //array of string yang menyimpan input dari pengguna
    vector<float> angkaKartu; //array of int yang menyimpan angka / nilai kartu
    int count; //angka penghitung
    string inputType; //jenis input: input dari pengguna atau random
    bool isInputValid = false; //validasi input 4 kartu dari pengguna
    bool isInputTypeValid = false; //validasi input random atau dari pengguna
    char saveSolution; //apakah ingin menyimpan solusi (y/n)
    std::string filename; //nama file text untuk menyimpan solusi
    bool isFilenameValid = false; //validasi input nama file
```

Program dimulai dengan tampilan awal berupa tampilan ASCII Art dari King of Clubs. ASCII Art ini dibuat dengan bantuan program ASCII Art Generator yang dapat diakses secara daring pada https://www.ascii-art-generator.org/.

```
  /* ALGORITMA */
  /* Output Title */
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWXOxONMMMMMMMMMMMMMMMMMMWd..kWNXk,.;xKMMMMMWX0kxxxk0NMMMMMNd..c0k;.:KMMMMMMMMMMMMMMMMMMMMMMMMMNxl0WMMMMMMMMMMMMMMM\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMXOOkxolc::coxONMMMWd..kWO;,kKc. ;KMW0o;..      .'cONMXl..xN0OKl.;KMMMMMMMMMMMMMMMMMMMMMMMMKxkXMMMMMMMMMMMMMMMMM\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWO;          'ckNO..xWN:  ,KX:  ;x:.            ,o: 'OWx..dK: :XMMMMWNNNNWMMMMMMMMMWko0WMMMMMMMMMMMMMMMMMM\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMWKOxdddxO0NWMMMMMMMMMMMMMMMMMNd.          .. ;XMNc  ,KM0o'       ..''..       .xWX;  :N0'.dWKxl;'.'',cxXMMMMXdoKMMMMMMMMMMMMMMMMX0NM\n";
  std::cout << "MMMMMMMMMMMMMMMMMMNkl,.       ..:xXMMMMMMMMMMMMMMMWO:ckxoc'      dWMMKc,xWMWNo    .cx0Kkx00x:.    :XMX;  .dWWl .:.          'xNMKodNMMMMMMMMMMMMMMMNO:.;k\n";
  std::cout << "MMMMMMMMMMMMMMMMMXx,               .dNMMMMMMMMMMMMMMMXocd0XKd,    :kxooooodxxdd:.  .:ddo;..,ldx:    cKNNOcxNMMx.     .,;,'.   .d0lxWMMMMMMMMMMMMMMMKdo:.'d\n";
  std::cout << "MMMMMMMMMMMMMMMWO,                  :KMMMMMMMMMMMMMMMNl..lodlllodxxxxxk0OOOOOOkk0OOOOOOOOxxxxxoooollddddxdoddx:    :kKWN0OOo.   .:OWMMMMMMMMMMMMMMMWOoxxc..k\n";
  std::cout << "MMMMMMMMMMMMMMMO'                    ;KMMMMMMMMMMMMMMMNocOK0Oxddllc:;::::::;;,'';::::::::llllodddxOOOOOK00OOkddlloooo:..ckk, .kWMMMMMMMMMMMMMMMXxdxxkl'.x\n";
  std::cout << "MMMMMMMMMMMMMWl                      .kMMMMMMMMMMMMMMMMK:..                              ...',;:loxxO0KXXKKOxdoo,.xWMMMMMMMMMMMMMMMWxoddxo,;:x\n";
  std::cout << "MMMMMMMMMMMMMWc                      .kMMMMMMMMMMMMMMMMWd    ......',,,,,,,,,,,,,,,,,,,,,'.....                ..',:lodk0OldWMMMMMMMMMMMMMMMMWd:xxxdcco0\n";
  std::cout << "MMMMMMMMMMMMMWx.                     ,KMMMMMMMMMMMMMMMMMk:dO0KXXXXXNWWWWWWWWWWWWWWWWWWWWWWXXXX0Ok:':'.,.....              ..cXMMMMMMMMMMMMMMMMWdcddd;;coX\n";
  std::cout << "MMMMMMMMMMMMMMNo.                    .kWMMMMMMMMMMMMMMMMMOdNMNNWMMMMMMMMMMMMMMMMMWNK0OxxxO0XWMMMMMXdOko0odxcx;;c,:''...     ;KMMMMMMMMMMMMMMMMMd;ddxcoodW\n";
  std::cout << "MMMMMMMMMMMMMMMWk,                   :0WMMMWMMMMMMMMMMMMMk;:lccccc:l0MMMMMMMMMXo;,,;;:cclooddkXMMMkd0oxOoOoxOckxxdodcl:c;.'dMMMMMMMMMMMMMMMMMMWd,oko;xxkM\n";
  std::cout << "MMMMMMMMN0OkkO0XWNx'               .dXXOdl:;;clx0NMMMMMMMk,l0XX0kdoxXMMMMMMMMMNOxk0KXXNWMMWN0kkKWMKoOddKlxkdKlokoxldlooooxlodKMMMMMMMMMMMMMMMMMo,ldd;oddW\n";
  std::cout << "MMMMMNkc'.     .;o0x.              .;;.        .;xNMMMMMk;lkO000000000NMMMMMMMXocdxkO0K00KWMMMWMMXldxlKooOlOddKokoodlxcdloodWMMMMMMMMMMMMMMMMMo.lxd:looX\n";
  std::cout << "MMMWk,           ..                              ;0MMMM0:colc:lllll::0MMMMMMMX;.,,':oooddoxNMMMMWdx0o0do0okxdKlxdcd:xldolxl0MMMMMMMMMMMMMMMMMo'lkO:;dd0\n";
  std::cout << "MMNd.                                             ;KMMMXc';:..ccd0kl;kMMMMMMNd;';c:l:;oo:..oKWMMMkkKd0OxKxOxc0xxdcOcxdoxcxlxMMMMMMMMMMMMMMMMx:kkdd:odk\n";
  std::cout << "MMk.                                              .xMMMX:':cdxdok0OolXMMMMMMN0OxoxOxoxOk0kkXWMMMMkkKlOOxKxOxcOoxxokcxdokcxxkMMMMMMMMMMMMMMMMO:lO0d;lxo\n";
  std::cout << "MWo                                                dMMMX:lKKXNWWNXKkKMMMMMMMMWKkO0OO0XWMMMMMMMMMMkk0:xxoKxOxcOlddlkcxkdkckxkMMMMMMMMMMMMMMMMMkck0d0k:dd\n";
  std::cout << "MWo                        .    ..                .xMMMKcdMWKO00OOxkWMMMMMMMMMMMN00OxxOXWMMMMMMMMMMxx0lkolKxOxcOlddcOcxdokckxkMMMMMMMMMMMMMMMMMOcokKOo;lO\n";
  std::cout << "MMk.                     .xXc   cXo               ,KMMM0lOMMWXNWW0kXMMMMMMWWNWMMMMMMMMMMMMMMMMMMMxx0d0dl0okkxOlOxd0cxolOckxkMMMMMMMMMMMMMMMMMk:kKxdKo,k\n";
  std::cout << "MMWd.                   .xWNc   :XXc             .kWMMMdcKMMMMMMKOXMMMXkOXOokNMMMMMMMMMMMMMMMMMMMxk0oKolOckddk:Oxx0l0ddkckxkMMMMMMMMMMMMMMMMMOldd0Oxo,d\n";
  std::cout << "MMMNd.                 .xNM0'   .OMNd.          .:0WMMMK:oWMMMMMM0x0XWMXOxdocdXMMMMMMMMMMMMMMMMMMNldxcKdoOd0ddk:OldOlKkkOo0xkMMMMMMMMMMMMMMMMMO;l0kd0k:x\n";
  std::cout << "MMMMW0c.             'l0WMWo     lWMWKd,.      .'cONMMMMWl'OMMMMMMMWXOOkKWMWKO0XMMMMMMMMMMMMMMMMMMMMNcdx:KxxOdKodkckloxcKkkOlOxkMMMMMMMMMMMMMMMMMMOoxdxKx.;k\n";
  std::cout << "MMMMMMWXkoc,,',:okXMMMM0'     .kMMMMWXOxxxkOKNMMMMMMMWO':XMMMMMMXx:c0WNWKl:oxONMMMMMMMMMMMMMMMMMMMNoxxcKxxOdKxxkckloOcOxd0lkooWMMMMMMMMMMMMMMMMO;c0Ooc,od\n";
  std::cout << "MMMMMMMMMMMMWWWWMMMMMMMMNc      ,0MMMMMMMMMMMMMMMMMMMXdcxNMMMMNx,.;dXMMMXxc;,,:OWMMMMMMMMMMMMMMMMMNldxc0dxOo0xx0oOdd0cxdc0dxxc0MMMMMMMMMMMMMMMMOlkxlkoldl\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMWd.       ,0MMMMMMMMMMMMMMMMMMWkxocKMMM0lcdKN0doddoo0NWXxodONMMMMMMMMMMMMMMMMNcoxc0xx0ckkdKdkOo0oo0lkklOllXMMMMMMMMMMMMMMMMO;:xk:'lld\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMWk.        ;XMMMMMMMMMMMWKOOKKdkc.kWMMXKNX0o,'...,,:oxkxkKXWMMMMMMMMMMMMMNKKodOl0xoKodOcxock:cc'l;,l:;;';lx0NMMMMMMMMMMMOcxxo:cdoK\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMNx:::::::::::l0MMMMMMMMMMNxoxko;dd,;xNMMMMWXOk00kkkOKXK00OKWMMMMMMMMMMMMMWXKdxxo0ld0cdl;c;',,,,,,::,;coddoolcddOWMMMMMMMMMO:ckk;ldxW\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMKlxKkdxkc;odOMMMMMMMMMW0dxxx0WMWMMMMMMMMMMMMMMMNXxxkdkokdlXo,xOkOOkdoooxdkklxkokxd0dkOoKMMMMMMMMMOcdko;odxM\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMW0odxxdx0O0dxMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWNOodldddxoOxolkKodkoOKdkdlKxkOl0dd0oOxxOdXMMMMMMMMMO;:kx,ldoN\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWKxdol::k0kOxkKKNMMMMMMMMMMMMMMMMMMMMMMMMMMMWW0lloddo:'oxd0x,oOdd0xdXdkxlKdxOoOodxod,;dKMMMMMMMMMk;lddlcxlk\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWOoOOxkO0d,lOxkdxxO00XXWNWMMMMMMMMMMMMMMMMMMMWNNXocd00000d;okdkkxkOOkxkddklkOodcll;:,colc:lx0XWMMMMMO,:O0o,odl\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMKcxKoxKdkk;xxxxdxdOxkxkk;dX0loXKkKNXWXXXkKkxXl:XKox0dxNo.;oxkkkkkOxdo,,c,'.     ;0WNo.   .;lxKNWklkkd0o,xo\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWKOdccc,oKxdOkolkxdkdkdxxdd;:.  ..  :l;ckxkkdKxd0od0O0KKNNooNx.  .;lloolc;.          :KMXl.        .';,'lO0dl:ld\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWKOkko.  .xXx':xkkkxxl:lcl:',....:.          ':,ollOdxOoo0OxkOddK0;                     :KMXc              'OOlkKl,d\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWKklco;:Kx.  dMO'  .:o;..'o;;l'      :o.            ..':.;o;':xkkkdc.                     cXMXc              .ld0Ooc;d\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWNKOx,  ,KkdXk.  oW0,..c0d.  dWxlkkddl'   :l.                                       cXMNl              .dKxl00lO\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMMMMMN0dcokKNl  .kKoOXc  .lOOkkxc.  ;KMN0OOOOOo.   :c.                                    cXMNl              :OodXK;:O\n";
  std::cout << "MMMMMMMMMMMMMMMMMMMMMMMWKxc'.   .';.   ;K0d0Kl.   ...   .oKXkollldkKXx.  .c:                                      .ld;  :XMNo.              .:OXkc'od\n";
  std::cout << "MMMMMMMMMMMMMMMMMMWKx:,;;,...         'ddlxX0o,.....;d00c.       .:kx.  .l;  ...                       'ol..dOc.;KMNd.              '00dkdlxc\n";
  std::cout << "MMMMMMMMMMMNOl,.   ....''''''..      ...'oOKK000XWWx.  .;oxxl'  .ol   ;d.,OXXo.                       .  .dWNc    ,0MWx.              'oo0O:dol\n\n";
  std::cout << "=====================================================================================================================================================\n\n";
  std::cout << "         $$$$$$   $$    $$         $$$$$$                                              $$$$$$            $$                                   \n";
```



Selanjutnya, program akan meminta input kepada pengguna. Terdapat dua jenis input yang dapat dimasukkan, yaitu input keempat kartu secara *custom* dari pengguna (opsi 1) atau input keempat kartu secara *random* atau acak (opsi 2). Jika pengguna memilih opsi 1, maka pengguna harus memasukkan input empat angka kartu yang dipisahkan dengan spasi (misalnya 8 J 3 A). Apabila input tidak sesuai, maka program akan terus menerus meminta input kepada pengguna hingga input sesuai. Setelah pengguna berhasil memasukkan empat angka kartu, maka program akan memasukkan keempat angka kartu ke vector kartu, lalu menerjemahkan angka kartu menjadi nilai float yang disimpan pada vector angkaKartu.

Jika pengguna memilih opsi 2, maka program akan memilih empat buah angka acak pada rentang 1 - 13. Empat angka acak ini disimpan dalam vector angkaKartu. Kemudian, program menerjemahkan nilai angka kartu (int) menjadi angka kartu (string) yang disimpan pada vector kartu.

```
/* INPUT */
while (!isInputTypeValid){
    std::cout << "Silakan pilih jenis input!\n1. Input dari pengguna\n2. Input random\n";
    getline(cin, inputType);
    if (inputType=="1"){
        /* Input dari pengguna */
        isInputTypeValid = true;
        while (!isInputValid){
            /* clear vector kartu */
            kartu.clear();
            angkaKartu.clear();
            /* Input 4 Kartu */
            std::cout << "\nSilakan masukkan 4 angka kartu [A,2,3,4,5,6,7,8,9,10,J,Q,K]!\n";
            getline(cin, inputKartu,'\n');
            /* Split string input */
            kartu = splitString(inputKartu);
            /* Ubah Input dari Bentuk String Menjadi Int, jika tidak sesuai maka tidak
dimasukkan ke vector angkaKartu */
            count = 0;
            while (count <=3 && charToFloat(kartu[count]) > 0){
                angkaKartu.push_back(charToFloat(kartu[count]));
                isInputValid = true;
                count++;
            }
```

```
            /* Validasi Input */
            if ((angkaKartu.size()==0) || (count <= 3 && charToFloat(kartu[count]) == 0 &&
count>0) || (kartu.size()>4)){
                /* input tidak valid ketika angka kartu yang dimasukkan tidak sesuai atau
angka kartu yang dimasukkan lebih dari 4 buah */
                std::cout << "Masukan tidak sesuai.\n";
                isInputValid = false;
            }
        }
    } else if (inputType=="2") {
        /* Input Random */
        isInputTypeValid = true;
        srand((unsigned) time(NULL));
        for (count=0; count<=3; count++){
            angkaKartu.push_back(rand() % 13 + 1);
        }
        for (count=0; count<=3; count++){
            kartu.push_back(intToChar(angkaKartu[count]));
        }
    } else {
        std::cout << "Masukan tidak sesuai.\n\n";
    }
}
```

Setelah berhasil melakukan input, program akan menampilkan kembali input angka kartu dan nilai yang sepadan dengan angka tersebut.

```
/* Cetak 4 angka kartu */
std::cout << "\nKartu: ";
for (count=0; count<=3; count++){
   std::cout << kartu[count] << " ";
}
std::cout << "\nAngka: ";
for (count=0; count<=3; count++){
    std::cout << angkaKartu[count] << " ";
}
std::cout << "\n";
```

Untuk menemukan solusi permainan, maka dilakukan proses dengan algoritma Brute Force. Prosedur brute_force yang telah direalisasikan sebelumnya dipanggil.

```
/* PROSES BRUTE FORCE */
auto start = std::chrono::system_clock::now(); //waktu awal eksekusi

brute_force(angkaKartu);
ln_operation = list_operation.size();
```

Seluruh solusi yang tersimpan pada list_operation dicetak ke layar beserta jumlahnya. Waktu yang diperlukan untuk melakukan proses tersebut juga turut dicetak ke layar.

```
/* OUTPUT */
std::cout << ln_operation << " solutions found\n";
for (count=0; count<=ln_operation-1; count++){
    cout << list_operation[count] << "\n";
}

auto end = std::chrono::system_clock::now(); //waktu akhir eksekusi
std::cout << "Execution Time: " << std::chrono::duration_cast<std::chrono::milliseconds>(end -
start).count() << "ms\n\n"; //output durasi eksekusi
```

Fitur terakhir pada program ini adalah fitur menyimpan hasil solusi permainan ke file teks. Pengguna akan ditanyakan apakah ingin menyimpan solusi atau tidak. Jika ya, pengguna dapat mengetikkan ‘y’, atau ‘n’ jika tidak. Apabila input pengguna salah (selain ‘y’ atau ‘n’), maka program akan terus meminta input hingga input sesuai. Jika pengguna menjawab ya, maka program akan meminta input nama file teks untuk menyimpan solusi dan mengecek apakah nama file sudah ada atau belum. Jika sudah terdapat file dengan nama yang sama dengan input pengguna, maka program akan meminta input kembali. Setelah input nama file sesuai, program akan menuliskan keempat angka kartu, jumlah solusi, dan seluruh operasi solusi permainan.

```
/* Simpan hasilnya ke text file */
isInputValid = false;
while (!isInputValid){
    std::cout << "Apakah Anda ingin menyimpan solusi? (y/n)\n";
    std::cin >> saveSolution;

    if (saveSolution=='y'){
       isInputValid = true;
       /* Simpan solusi */
       while (!isFilenameValid){
            std::cout << "Masukkan nama file untuk menyimpan solusi!\n";
            std::cin >> filename;
            ofstream outfile;
            outfile.open("../test/" + filename + ".txt", ios_base::out | ios_base::in);
            if (!outfile.is_open()){
                isFilenameValid = true;
                outfile.open("../test/" + filename + ".txt");
                outfile << "Kartu: " << kartu[0] << " " << kartu[1] << " " << kartu[2] << " "
<< kartu[3] << endl;
                outfile << ln_operation << " solutions found" << endl;
                for (count=0; count<=ln_operation-1; count++){
                    outfile << list_operation[count] << endl;
                }

                std::cout << "File " << filename << ".txt berhasil tersimpan!\n";
            } else {
```

```
                std::cout << filename << ".txt sudah ada. Silakan masukkan kembali nama file
yang berbeda!\n\n";
            }
            outfile.close();
        }
    } else if (saveSolution=='n'){
        isInputValid = true;
    } else {
        isInputValid = false;
        std::cout << "Masukan tidak sesuai.\n\n";
    }
}
```

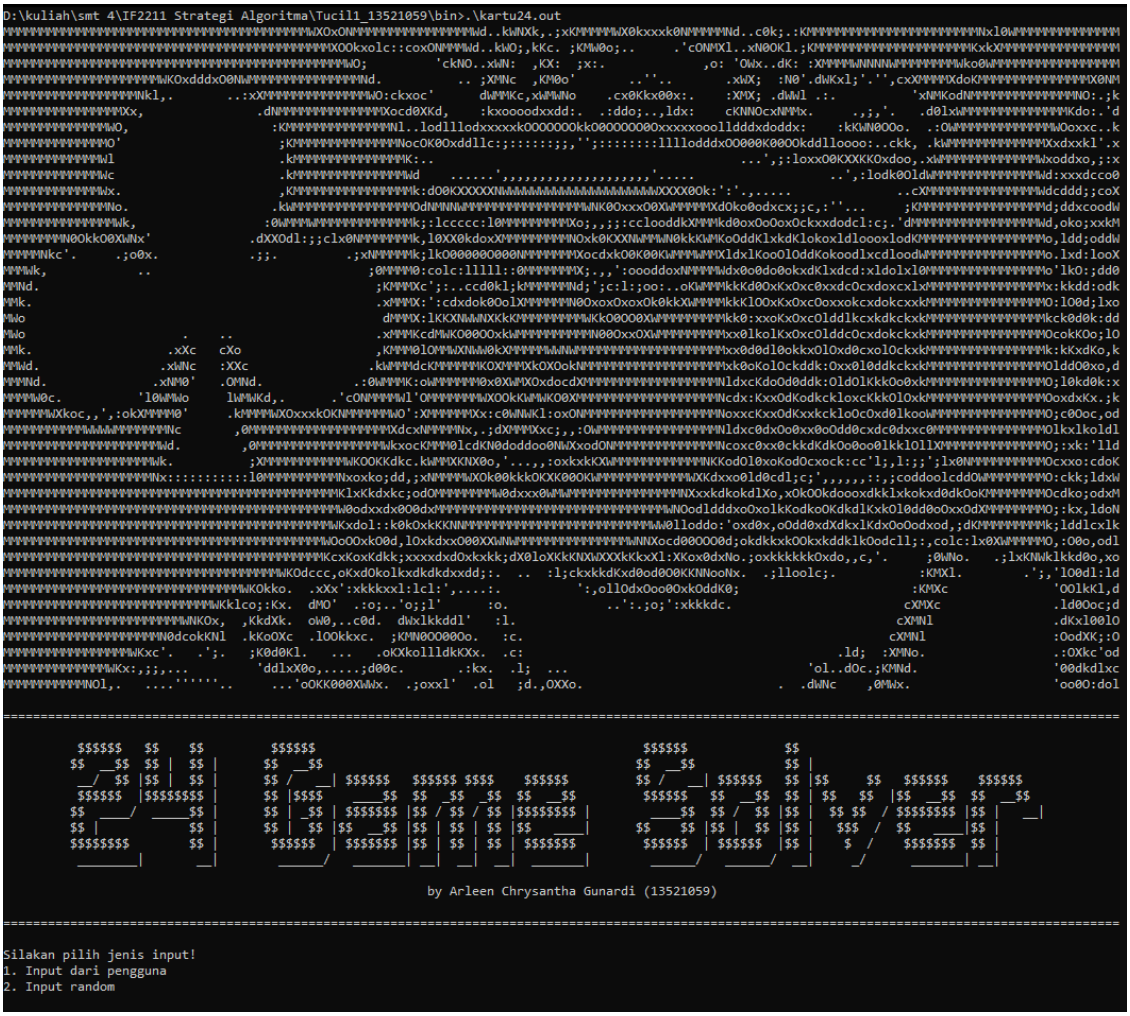
Akhirnya, program selesai dengan menampilkan pesan tampilan program selesai.

```
    std::cout << "========================================================================= Program
selesai ==========================================================================\n\n";

    return 0;
}
```

IV. Hasil Eksekusi Program.

4.1 Tampilan Awal

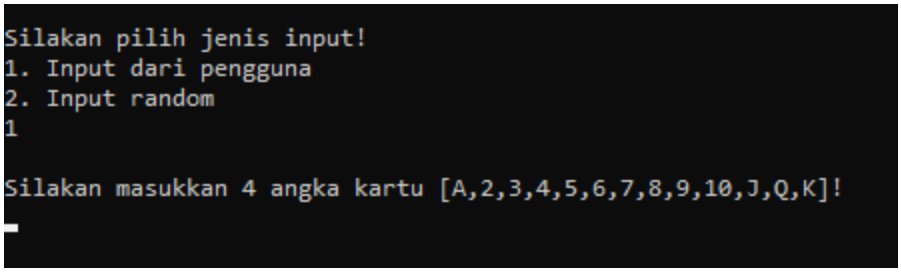
Program dapat dijalankan dengan menjalankan perintah “./kartu24.out” pada terminal yang berada pada directory folder bin. Berikut adalah tampilan awal program.



Gambar 4.1.1 Tampilan Awal Program

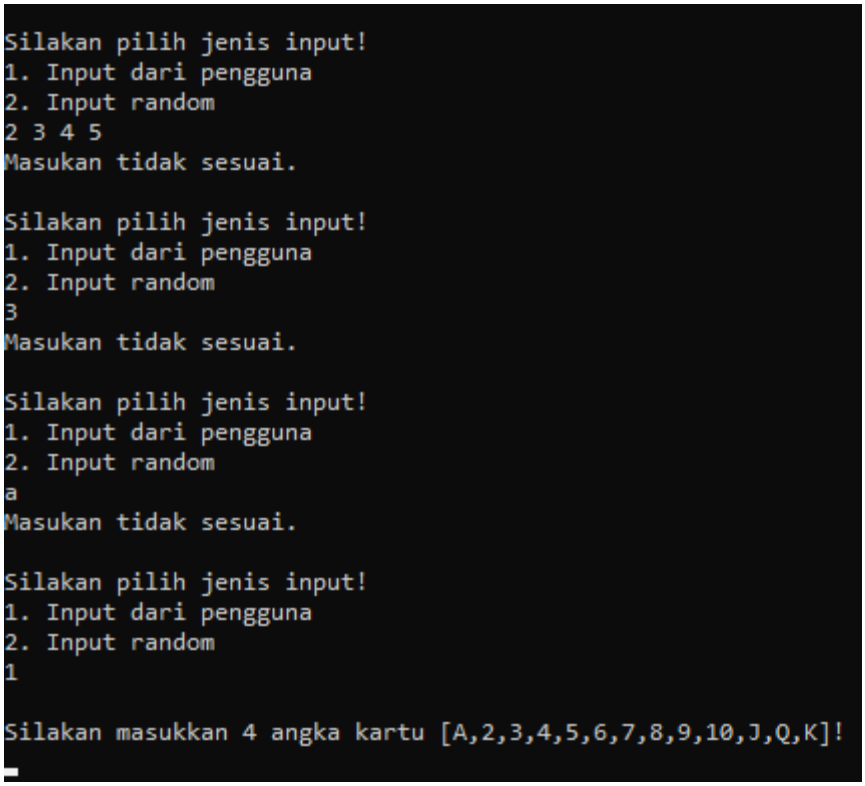
4.2 Jenis Input

Pada saat program dijalankan, pengguna diminta untuk memilih jenis input angka kartu. Ketika pengguna memilih 1, maka pengguna akan diminta untuk memasukkan empat buah angka. Namun, jika pengguna memilih 2, maka program langsung mencetak angka kartu beserta solusinya. Apabila input yang dimasukkan bukan 1 atau 2, maka program akan meminta input kembali hingga input valid. Berikut adalah tampilan ketika pengguna memilih jenis input yang ingin dimasukkan.



Gambar 4.2.1 Pemilihan Jenis Input dari Pengguna

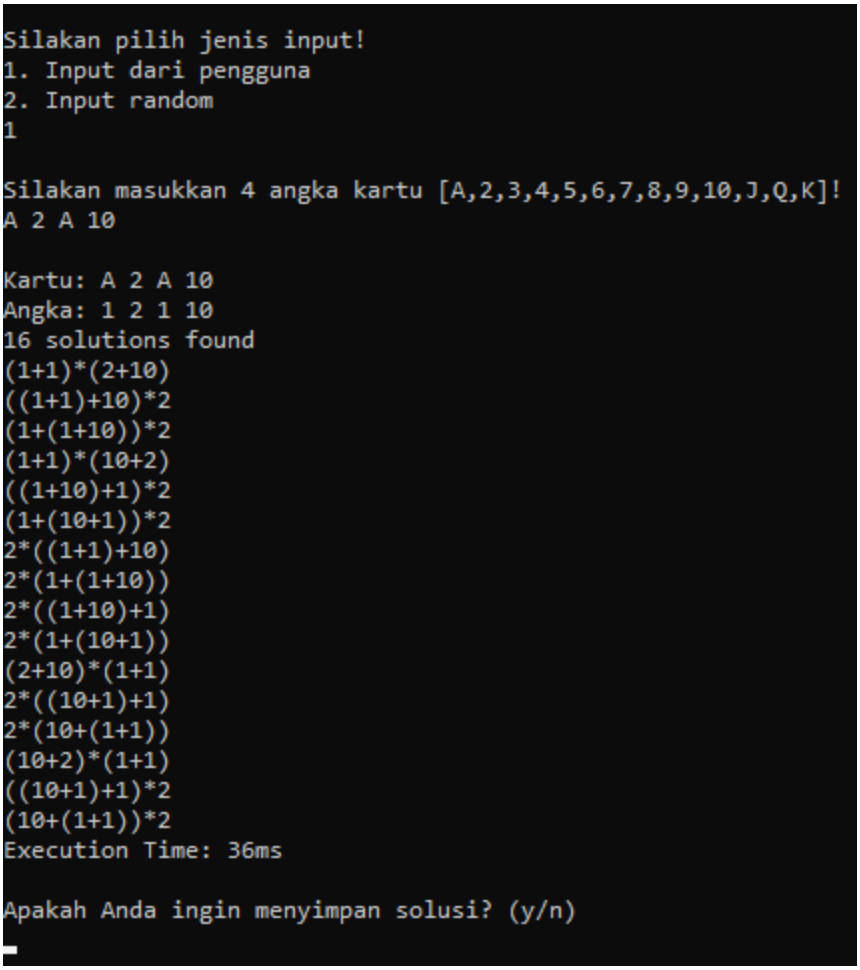
Gambar 4.2.2 Pemilihan Jenis Input Random



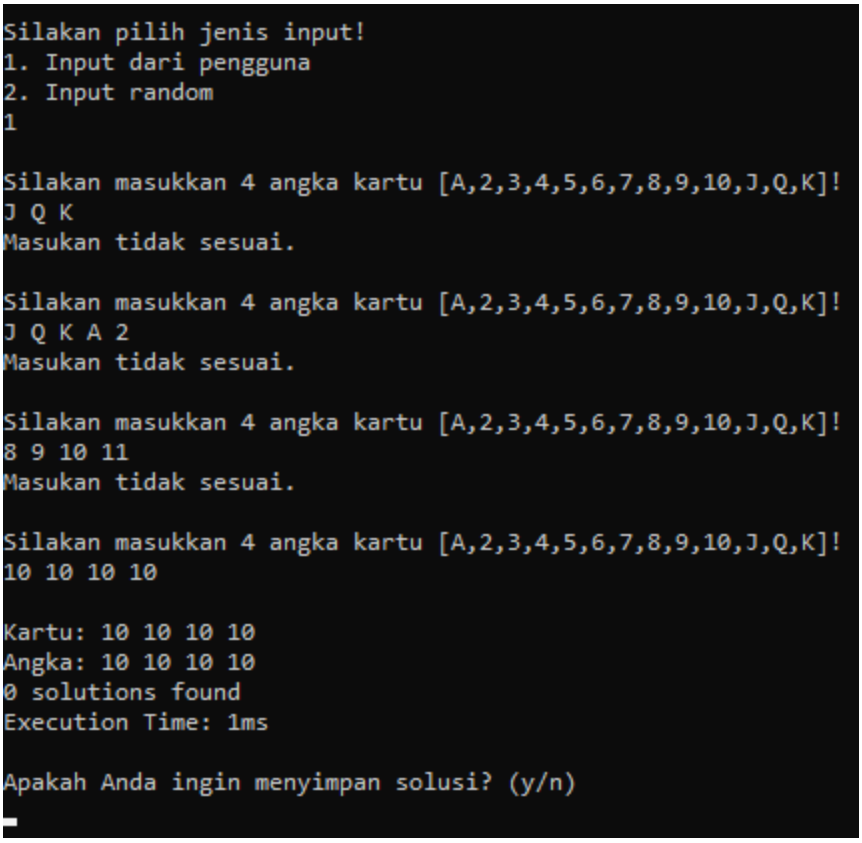
Gambar 4.2.3 Validasi Jenis Input

4.3 Input dari Pengguna dan Solusinya

Ketika memilih jenis input yang pertama, pengguna dapat memasukkan empat angka kartu [A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K] yang dipisahkan oleh spasi, misalnya “A 2 A 10”. Jika masukan pengguna salah, maka program akan terus meminta input hingga input valid. Setelah itu, program akan menampilkan angka kartu beserta dengan nilai yang setara dengan angka kartu tersebut. Jumlah solusi ditampilkan beserta dengan seluruh solusi permainan berdasarkan empat buah angka masukan dari pengguna. Durasi waktu eksekusi program juga ditampilkan setelah program menampilkan semua solusi permainan. Berikut adalah tampilan ketika pengguna memilih untuk memasukkan input dari pengguna.



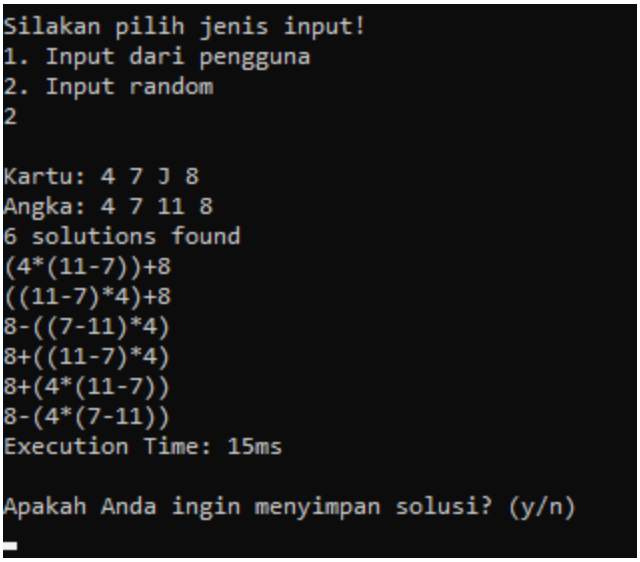
Gambar 4.3.1 Solusi Berdasarkan Input dari Pengguna



Gambar 4.3.2 Validasi Input Angka Kartu

4.4 Input Acak dan Solusinya

Jika pengguna memilih jenis input kedua, maka program akan menampilkan empat angka kartu yang acak beserta nilai yang sepadan. Program juga langsung mencetak seluruh solusi permainan berdasarkan empat angka acak tersebut. Waktu eksekusi juga ditampilkan pada akhir program. Berikut adalah tampilan ketika pengguna memilih untuk mendapatkan angka acak.



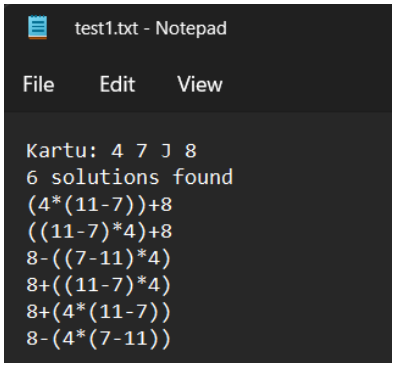
Gambar 4.4.1 Solusi Berdasarkan Input Random

4.5 Simpan Solusi ke File

Setelah program menampilkan solusi permainan, pengguna dapat memilih untuk menyimpan solusi tersebut ke dalam suatu file teks. Apabila pengguna tidak ingin menyimpan solusi, pengguna dapat menmasukkan perintah ‘n’ dan program selesai. Namun, jika pengguna ingin menyimpannya, pengguna dapat memasukkan perintah ‘y’. Setelah itu, program akan meminta nama file untuk menyimpan solusi tersebut. Apabila nama file masukan dari pengguna telah ada pada folder test, maka program akan meminta kembali nama file hingga input valid. Kemudian, file teks berisi solusi permainan selesai dibuat dan program selesai. File teks berisi solusi tersebut dapat diakses pada folder test. Berikut adalah tampilan ketika pengguna ingin menyimpan solusi ke file teks.

```
Silakan pilih jenis input!
1. Input dari pengguna
2. Input random
2

Kartu: 4 7 J 8
Angka: 4 7 11 8
6 solutions found
(4*(11-7))+8
((11-7)*4)+8
8-((7-11)*4)
8+((11-7)*4)
8+(4*(11-7))
8-(4*(7-11))
Execution Time: 15ms

Apakah Anda ingin menyimpan solusi? (y/n)
y
Masukkan nama file untuk menyimpan solusi!
test1
File test1.txt berhasil tersimpan!
============================================================ Program selesai ============================================================

D:\kuliah\smt 4\IF2211 Strategi Algoritma\Tucil1_13521059\bin>
```

Gambar 4.5.1 Menyimpan Solusi ke File Teks



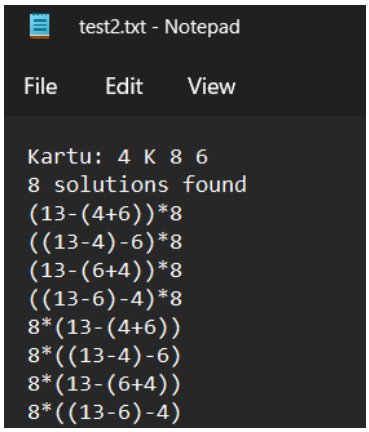
Gambar 4.5.2 Hasil File Teks test1

```
(8+3)+(11+2)
8+((3+11)+2)
8+(3+(11+2))
8-((3-11)*2)
(8/3)*(11-2)
8/(3/(11-2))
Execution Time: 259ms

Apakah Anda ingin menyimpan solusi? (y/n)
n
============================================================ Program selesai ============================================================

D:\kuliah\smt 4\IF2211 Strategi Algoritma\Tucil1_13521059\bin>
```

Gambar 4.5.3 Solusi Tidak Disimpan

```
Kartu: 4 K 8 6
Angka: 4 13 8 6
8 solutions found
(13-(4+6))*8
((13-4)-6)*8
(13-(6+4))*8
((13-6)-4)*8
8*(13-(4+6))
8*((13-4)-6)
8*(13-(6+4))
8*((13-6)-4)
Execution Time: 14ms

Apakah Anda ingin menyimpan solusi? (y/n)
g
Masukan tidak sesuai.

Apakah Anda ingin menyimpan solusi? (y/n)
y
Masukkan nama file untuk menyimpan solusi!
test1
test1.txt sudah ada. Silakan masukkan kembali nama file yang berbeda!

Masukkan nama file untuk menyimpan solusi!
test2
File test2.txt berhasil tersimpan!
============================================================ Program selesai ================================================================


D:\kuliah\smt 4\IF2211 Strategi Algoritma\Tucil1_13521059\bin>
```

Gambar 4.5.4 Validasi Input Simpan Solusi dan Nama File

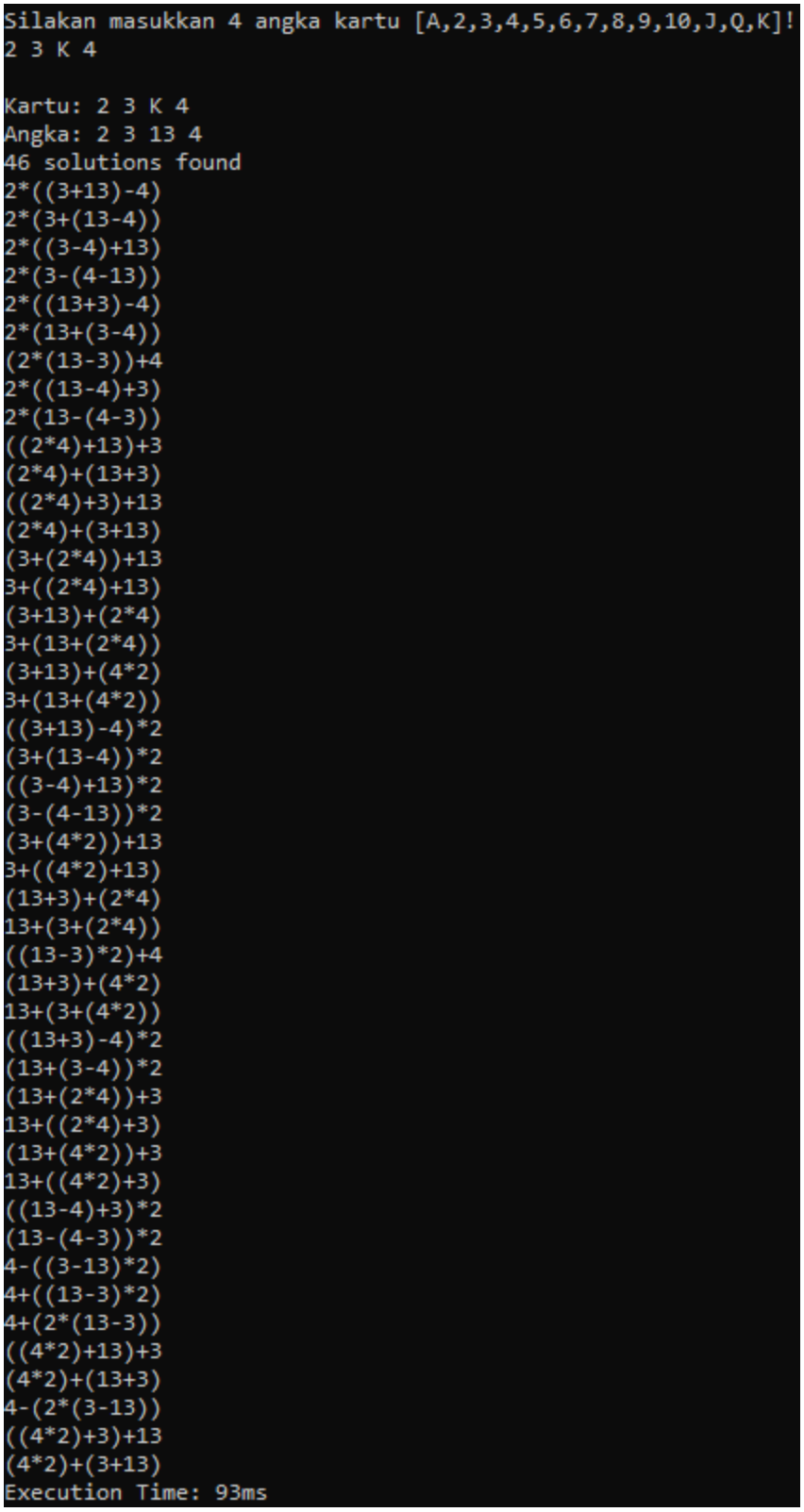


Gambar 4.5.2 Hasil File Teks test2

4.6 Contoh Solusi

4.6.1 Test Case 1

Test case untuk kartu 2 3 K 4.



Gambar 4.6.1 Hasil Test Case 1

4.6.2 Test Case 2

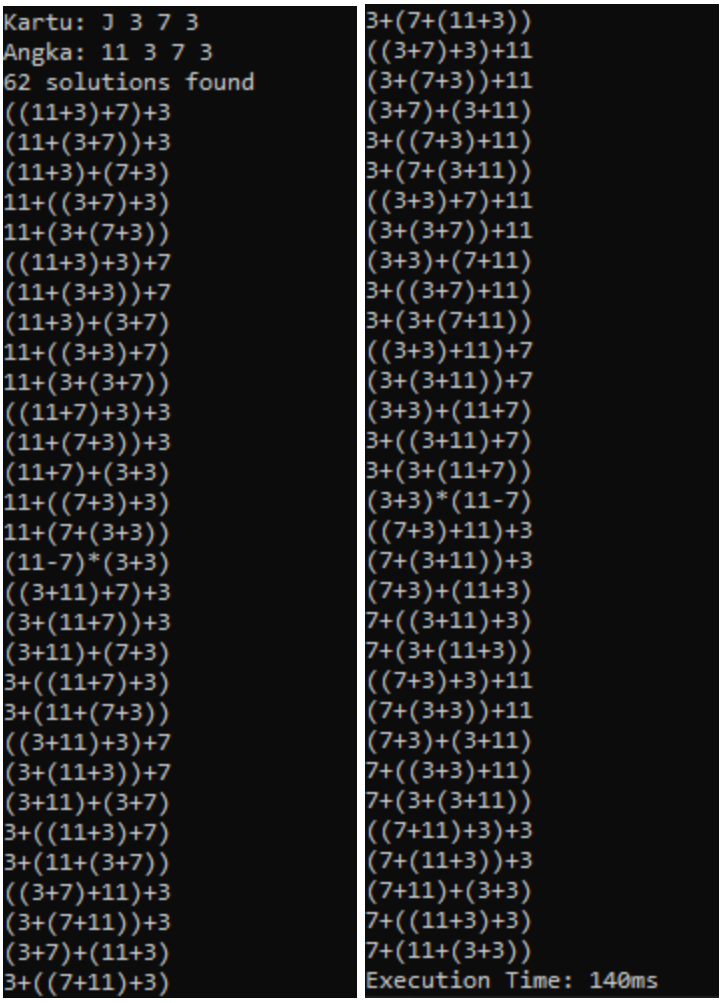
Test case untuk kartu 6 7 J 8.

```
Silakan masukkan 4 angka kartu [A,2,3,4,5,6,7,8,9,10,J,Q,K]!
6 7 J 8

Kartu: 6 7 J 8
Angka: 6 7 11 8
32 solutions found
6*((7-11)+8)
6*(7-(11-8))
6*((7+8)-11)
6*(7+(8-11))
6*((8-11)+7)
6*(8-(11-7))
6*((8+7)-11)
6*(8+(7-11))
((7+11)/6)*8
(7+11)/(6/8)
((7+11)*8)/6
(7+11)*(8/6)
((7-11)+8)*6
(7-(11-8))*6
((7+8)-11)*6
(7+(8-11))*6
((11+7)/6)*8
(11+7)/(6/8)
((11+7)*8)/6
(11+7)*(8/6)
((8+7)-11)*6
(8+(7-11))*6
(8*(7+11))/6
8*((7+11)/6)
((8-11)+7)*6
(8-(11-7))*6
(8*(11+7))/6
8*((11+7)/6)
(8/6)*(11+7)
8/(6/(11+7))
(8/6)*(7+11)
8/(6/(7+11))
Execution Time: 62ms
```

Gambar 4.6.2 Hasil Test Case 2

4.6.3 Test Case 3

Test case untuk kartu J 3 7 3.



Gambar 4.6.3 Hasil Test Case 3

4.6.4 Test Case 4

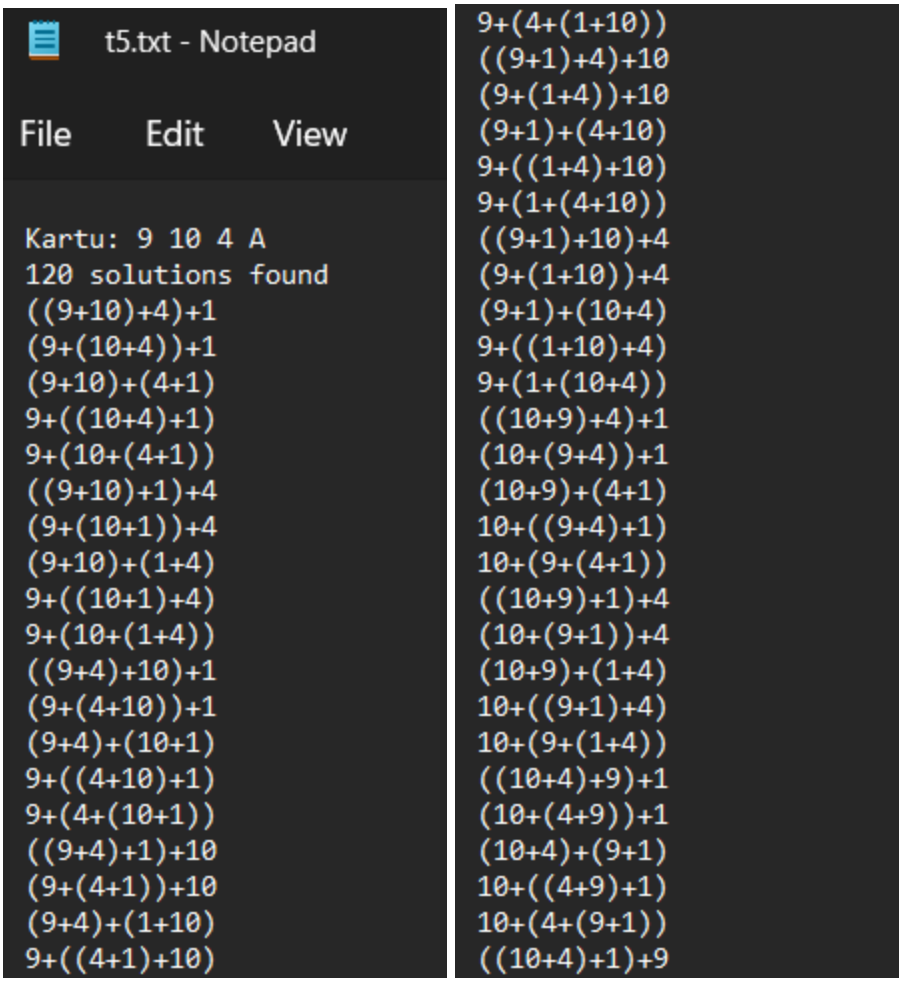
Test case untuk kartu 5 8 Q 8.

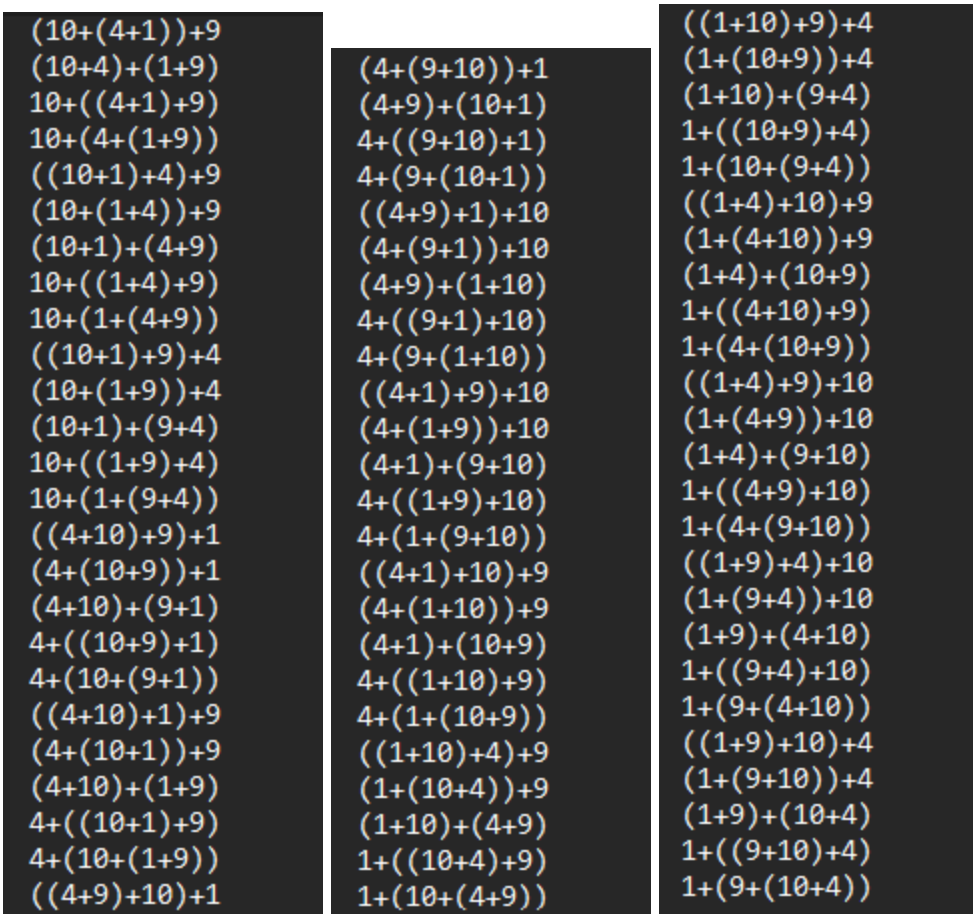
```
Silakan masukkan 4 angka kartu [A,2,3,4,5,6,7,8,9,10,J,Q,K]!
5 8 Q 8

Kartu: 5 8 Q 8
Angka: 5 8 12 8
0 solutions found
Execution Time: 0ms
```

Gambar 4.6.4 Hasil Test Case 4

4.6.5 Test Case 5

Test case untuk kartu 9 10 4 A.


t5.txt - Notepad
File Edit View
Kartu: 9 10 4 A
120 solutions found
((9+10)+4)+1
(9+(10+4))+1
(9+10)+(4+1)
9+((10+4)+1)
9+(10+(4+1))
((9+10)+1)+4
(9+(10+1))+4
(9+10)+(1+4)
9+((10+1)+4)
9+(10+(1+4))
((9+4)+10)+1
(9+(4+10))+1
(9+4)+(10+1)
9+((4+10)+1)
9+(4+(10+1))
((9+4)+1)+10
(9+(4+1))+10
(9+4)+(1+10)
9+((4+1)+10)
9+(4+(1+10))
((9+1)+4)+10
(9+(1+4))+10
(9+1)+(4+10)
9+((1+4)+10)
9+(1+(4+10))
((9+1)+10)+4
(9+(1+10))+4
(9+1)+(10+4)
9+((1+10)+4)
9+(1+(10+4))
((10+9)+4)+1
(10+(9+4))+1
(10+9)+(4+1)
10+((9+4)+1)
10+(9+(4+1))
((10+9)+1)+4
(10+(9+1))+4
(10+9)+(1+4)
10+((9+1)+4)
10+(9+(1+4))
((10+4)+9)+1
(10+(4+9))+1
(10+4)+(9+1)
10+((4+9)+1)
10+(4+(9+1))
((10+4)+1)+9



Gambar 4.6.5 Hasil Test Case 5

4.6.6 Test Case 6

Test case untuk kartu Q 8 K 10.

```
Silakan masukkan 4 angka kartu [A,2,3,4,5,6,7,8,9,10,J,Q,K]!
Q 8 K 10

Kartu: Q 8 K 10
Angka: 12 8 13 10
8 solutions found
(12/(13-8))*10
12/((13-8)/10)
(12*10)/(13-8)
12*(10/(13-8))
(10/(13-8))*12
10/((13-8)/12)
(10*12)/(13-8)
10*(12/(13-8))
Execution Time: 15ms
```

Gambar 4.6.6 Hasil Test Case 6

V. Lampiran

5.1 Tautan Repository GitHub

Berikut adalah tautan repository GitHub untuk program.

https://github.com/arleenchr/Tucil1_13521059

5.2 Tabel Penilaian

Berikut adalah tabel untuk penilaian.

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		

VI. Daftar Referensi

24 Solver. http://24solver.us-west-2.elasticbeanstalk.com/ (Diakses 23 Januari 2023)

Chandra, Evita. 2015. "Penerapan Algoritma Brute Force pada Permainan Kartu 24 (24 *game*)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf (Diakses 23 Januari 2023)

LeetCode. 2017. "Short Permutation in a Long String". https://leetcode.com/problems/permutation-in-string/solutions/127729/official-solution/ (Diakses 20 Januari 2023)

Munir, Rinaldi. 2023. "Algoritma Brute Force (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/stima22-23.htm (Diakses 23 Januari 2023)