Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*

Bernardus Willson
K03 / 13521021

1. Algoritma Bruteforce

Algoritma brute force secara singkat ialah cara penyelesaian masalah dengan program yang meninjau semua kasus yang mungkin muncul berdasarkan deskripsi masalah yang diberikan.

1.1 Algoritma Permutasi

```
int save[1000][4] = {};
int count = 0;
int col = 0;
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        for (int k = 0; k < 4; k++) {
            for (int l = 0; l < 4; l++) {
                if (i != j && i != k && i != l && j != k && j != l && k != l) {
                    if (i == 0) {
                        save[count][col] = A;
                        col++;
                    }
                    if (i == 1) {
                        save[count][col] = B;
                        col++;
                    }
                    if (i == 2) {
                        save[count][col] = C;
                        col++;
                    }
                    if (i == 3) {
                        save[count][col] = D;
                        col++;
                    }
                    if (j == 0) {
                        save[count][col] = A;
                        col++;
                    }
                    if (j == 1) {
                        save[count][col] = B;
                        col++;
                    }
                    if (j == 2) {
                        save[count][col] = C;
                        col++;
                    }
```

```
                    if (j == 3) {
                        save[count][col] = D;
                        col++;
                    }
                    if (k == 0) {
                        save[count][col] = A;
                        col++;
                    }
                    if (k == 1) {
                        save[count][col] = B;
                        col++;
                    }
                    if (k == 2) {
                        save[count][col] = C;
                        col++;
                    }
                    if (k == 3) {
                        save[count][col] = D;
                        col++;
                    }
                    if (l == 0) {
                        save[count][col] = A;
                        col = 0;
                    }
                    if (l == 1) {
                        save[count][col] = B;
                        col = 0;
                    }
                    if (l == 2) {
                        save[count][col] = C;
                        col = 0;
                    }
                    if (l == 3) {
                        save[count][col] = D;
                        col = 0;
                    }
                    count++;
                }
            }
        }
    }
}
int resultCount = 1;
int resultCounts = 0;
int saveCount = 1;
result[0][0] = save[0][0];
result[0][1] = save[0][1];
result[0][2] = save[0][2];
```

```
    result[0][3] = save[0][3];
    while (saveCount < count) {
        for (int x = 0; x < resultCount; x++) {
            if (result[x][0] == save[saveCount][0] && result[x][1] == save[saveCount][1] &&
result[x][2] == save[saveCount][2] && result[x][3] == save[saveCount][3]) {
                break;
            }
            else {
                resultCounts++;
            }
            if (resultCounts == resultCount) {
                result[resultCount][0] = save[saveCount][0];
                result[resultCount][1] = save[saveCount][1];
                result[resultCount][2] = save[saveCount][2];
                result[resultCount][3] = save[saveCount][3];
                resultCount++;
            }
        }
        resultCounts = 0;
        saveCount++;
    }
    *counter = resultCount;
```

Pada algoritma di atas, penulis memasukkan algoritma tersebut ke dalam suatu prosedur bernama “permutation”, dengan parameter input 4 angka integer yang merupakan angka dari kartu-kartu yang diperoleh, dan dengan output berupa matrix of integer yang nantinya akan menyimpan hasil dari permutasi angka-angka, serta counter sebagai penanda ada berapa banyak permutasi yang diperoleh.

Di sini, penulis menggunakan pendekatan *brute force* sebagaimana telah ditunjukkan di atas. Pada dasarnya, penulis menggunakan nested loop sebanyak 4 kali, yaitu sebanyak jumlah kartu yang dibutuhkan. Loop-loop tersebut terdiri dari i, j, k, l yang dapat bernilai dari 0 sampai 3, kemudian diberi kondisi agar angka-angka yang didapat dari i, j, k, l tidak sama. Secara logika, loop tersebut akan menghasilkan semua kombinasi angka yang mungkin seperti misalnya 0 1 2 3, 0 1 3 2, dan seterusnya. Setiap angka tersebut memiliki arti: angka 0 berarti kartu pertama, angka 1 berarti kartu kedua, angka 2 berarti kartu ketiga, dan angka 3 berarti kartu keempat. Maka dari itu akan didapat semua kombinasi posisi angka kartu yang mungkin, dan kemudian angka-angka kartu tersebut disimpan ke dalam matrix “save”.

Namun bagaimana jika terdapat kartu yang angkanya sama? Misalnya 2 3 3 5. Tentunya loop tersebut akan menghasilkan beberapa angka yang masih duplikat, sedangkan yang kita mau hanya permutasinya saja, dengan kata lain angka-angka yang masih duplikat tersebut harus dihilangkan. Caranya adalah dengan membuat matrix baru yang masih kosong bernama “result”, lalu setiap baris dari matrix “save” dipindahkan ke matrix baru tersebut dengan syarat tidak ada angka-angka dari 1 baris tersebut yang sama. Maka dari itu dibutuhkan

algoritma searching terhadap matrix baru tersebut, jika angka-angka tersebut belum ada di matrix baru, maka angka-angka yang ada di 1 baris matrix “save” tersebut dipindahkan ke matrix ”result”. Jika pada saat proses searching ternyata ditemukan bahwa angka-angka tersebut sudah ada di matrix baru, maka proses searching tersebut secara otomatis berhenti, lalu kemudian akan dilakukan proses searching baru untuk baris selanjutnya di matrix “save”.

1.2 **Algoritma Operasi dan Kurung**

```
     for (int h = 0; h < row; h++) {
        for (int i = 1; i < 5; i++) {
            for (int j = 1; j < 5; j++ ) {
                for (int k = 1; k < 5; k++) {
                    // ((A _ B) _ C) _ D
                    if (operators(operators(operators(permRes[h][0], permRes[h][1], i),
permRes[h][2], j), permRes[h][3], k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = "(("+Card1+" "+op1+" "+Card2+") "+op2+" "+Card3+") "+op3+"
"+Card4;
                        count++;
                    }
                    //  (A _ (B _ C)) _ D
                    if (operators(operators(permRes[h][0], operators(permRes[h][1], permRes[h][2],
i), j), permRes[h][3], k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = "("+Card1+" "+op2+" ("+Card2+" "+op1+" "+Card3+")) "+op3+"
"+Card4;
                        count++;
                    }
                    // (A _ B) _ (C _ D)
                    if (operators(operators(permRes[h][0], permRes[h][1], i),
operators(permRes[h][2], permRes[h][3], j), k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = "("+Card1+" "+op1+" "+Card2+") "+op3+" ("+Card3+" "+op2+"
"+Card4+")";
                        count++;
                    }
```

```
                    // A _ ((B _ C) _ D)
                    if (operators(permRes[h][0], operators(operators(permRes[h][1], permRes[h][2],
i), permRes[h][3], j), k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = Card1+" "+op3+" (("+Card2+" "+op1+" "+Card3+") "+op2+"
"+Card4+")";
                        count++;
                    }
                    // A _ (B _ (C _ D))
                    if (operators(permRes[h][0], operators(permRes[h][1], operators(permRes[h][2],
permRes[h][3], i), j), k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = ""+Card1+" "+op3+" ("+Card2+" "+op2+" ("+Card3+" "+op1+"
"+Card4+"))";
                        count++;
                    }
                }
            }
        }
    }
```

Pada algoritma di atas, penulis tidak meletakkan algoritma tersebut ke dalam fungsi ataupun prosedur, letak algoritma ini berada di main.cpp.

Di sini, penulis menggunakan pendekatan *brute force* sebagaimana telah ditunjukkan di atas. Pada dasarnya, penulis menggunakan nested loop sebanyak 4 kali, yaitu 1 loop untuk mengakses baris dari matrix hasil permutasi, serta 3 loop sebanyak jumlah operasi yang dibutuhkan. Loop-loop tersebut terdiri dari h, i, j, k, dimana h berhenti berjalan ketika seluruh elemen matriks permutasi sudah diakses, sedangkan i, j, k bernilai 1 sampai 5 yang nantinya akan dipakai di sebuah fungsi bernama “operators” dengan algoritma sebagai berikut.

```
    double result;
    if (n==1) {
        result = a + b;
    }
    else if (n==2) {
        result = a - b;
    }
    else if (n==3) {
        result = a * b;
```

```
    }
    else if (n==4) {
        result = a / b;
    }
    return result;
```

Fungsi tersebut terdiri dari beberapa parameter, yaitu 2 angka masukan sebagai angka-angka yang ingin dioperasikan, serta 1 angka sebagai penentu operasi apa yang akan dipakai. Angka 1 berarti operasi penjumlahan, angka 2 berarti operasi pengurangan, angka 3 berarti operasi perkalian, dan angka 4 berarti operasi pembagian. Fungsi kemudian akan mengembalikan hasil dari pengoperasian kedua angka masukan.

Angka-angka yang dihasilkan oleh i, j, k berfungsi sebagai penanda operasi apa yang akan digunakan. Maka dari itu terbentuklah semua kemungkinan operasi-operasi yang akan digunakan. Semua kemungkinan operasi tersebut diaplikasikan ke setiap hasil permutasi yang telah didapat.

Tidak lupa untuk mem-filter semua hasil yang didapat dari pengoprasian angka-angka, karena yang diminta hanyalah yang hasilnya adalah 24. Karena penulis hanya menggunakan 2 angka dalam fungsi operasi, maka penulis mendapatkan 5 kondisi berbeda dengan mengaplikasikan fungsi operasi di dalam fungsi operasi. Dengan ini, maka kita tidak perlu memikirkan penggunaan kurung, karena kita bisa memilih angka mana yang ingin dioperasikan duluan. 5 kondisi tersebut di antaranya: ((A _ B) _ C) D, (A _ (B _ C) _ D, (A _ B) _ (C _ D), A _ ((B _ C) _ D), dan A _ (B _ (C _ D)). Kemudian hasil yang bernilai 24 dihitung ada berapa solusi, dan solusi-solusinya disimpan ke dalam matrix of string bernama “saveRes” karena di spek tucil diminta untuk menampilkan jumlah solusinya terlebih dahulu baru menampilkan solusi-solusinya.

2. Source Code Program

2.1 **main.cpp**

```
#include <iostream>
#include <string>
#include <chrono>
#include "others\inputCard.cpp"
#include "algorithm\operators.cpp"
#include "algorithm\numPermutation.cpp"
#include "others\saveDat.cpp"
#include "others\splashScreen.cpp"

using namespace std;

int main() {
    int input;
    int CardInt[4];
    bool flag = false;
    using Clock = chrono::high_resolution_clock;
```

```
    welcomeScreen();
    while (flag == false) {
        menu();
        cin >> input;
        if (input == 1) {
            inputTerminal(CardInt);
            flag = true;
        }
        else if (input == 2) {
            inputRandom(CardInt);
            flag = true;
        }
        else if (input == 3) {
            help();
            flag = false;
        }
        else if (input == 4) {
            exit(0);
        }
        else {
            cout << "!MASUKAN TIDAK SESUAI, MOHON ULANGI MASUKAN!" << endl;
            flag = false;
        }
    }
    auto start = Clock::now();
    string saveRes[10000] = {};
    int row = 0;
    int count = 0;
    string op1, op2, op3;
    int permRes[24][4];
    permutation(CardInt[0], CardInt[1], CardInt[2], CardInt[3], permRes, &row);
    for (int h = 0; h < row; h++) {
        for (int i = 1; i < 5; i++) {
            for (int j = 1; j < 5; j++ ) {
                for (int k = 1; k < 5; k++) {
                    // ((A _ B) _ C) _ D
                    if (operators(operators(operators(permRes[h][0], permRes[h][1], i),
permRes[h][2], j), permRes[h][3], k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = "(("+Card1+" "+op1+" "+Card2+") "+op2+" "+Card3+")
"+op3+" "+Card4;
                        count++;
                    }
                    //  (A _ (B _ C)) _ D
```

```
                    if (operators(operators(permRes[h][0], operators(permRes[h][1], permRes[h][2],
i), j), permRes[h][3], k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = "("+Card1+" "+op2+" ("+Card2+" "+op1+" "+Card3+"))
"+op3+" "+Card4;
                        count++;
                    }
                    // (A _ B) _ (C _ D)
                    if (operators(operators(permRes[h][0], permRes[h][1], i),
operators(permRes[h][2], permRes[h][3], j), k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = "("+Card1+" "+op1+" "+Card2+") "+op3+" ("+Card3+" "+op2+"
"+Card4+")";
                        count++;
                    }
                    // A _ ((B _ C) _ D)
                    if (operators(permRes[h][0], operators(operators(permRes[h][1], permRes[h][2],
i), permRes[h][3], j), k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = Card1+" "+op3+" (("+Card2+" "+op1+" "+Card3+") "+op2+"
"+Card4+")";
                        count++;
                    }
                    // A _ (B _ (C _ D))
                    if (operators(permRes[h][0], operators(permRes[h][1], operators(permRes[h][2],
permRes[h][3], i), j), k) == 24) {
                        string Card1 = to_string(permRes[h][0]);
                        string Card2 = to_string(permRes[h][1]);
                        string Card3 = to_string(permRes[h][2]);
                        string Card4 = to_string(permRes[h][3]);
                        opOutput(i, j, k, &op1, &op2, &op3);
                        saveRes[count] = ""+Card1+" "+op3+" ("+Card2+" "+op2+" ("+Card3+" "+op1+"
"+Card4+"))";
                        count++;
                    }
                }
```

```
            }
        }
    }
    cout << endl;
    if (count == 0) {
        cout << "==================" << endl;
        cout << "Tidak ada solusi!" << endl;
    }
    else {
        cout << "====================" << endl;
        cout << count << " SOLUSI DITEMUKAN!"<< endl;
        cout << "====================" << endl;
        cout << "Solusi: " << endl;
        for (int i = 0; i < count; i++) {
            cout << saveRes[i] << endl;
        }
    }
    auto end = Clock::now();
    auto ms = chrono::duration_cast<chrono::milliseconds>(end - start).count();
    cout << "================================" << endl;
    cout << "Execution time: " << ms << " milliseconds" << endl;
    cout << "================================" << endl;
    cout << endl;
    save(saveRes, count, CardInt);
    string input1;
    bool flag1 = false;
    while (flag1 == false) {
        cout << endl;
        cout << "Apakah Anda ingin mencoba angka lain? (y/n): ";
        cin >> input1;
        if (input1 == "y") {
            flag1 = true;
            main();
        }
        else if (input1 == "n") {
            exit(0);
        }
        else {
            cout << "!MASUKAN TIDAK SESUAI, MOHON ULANGI MASUKAN!" << endl;
            flag1 = false;
        }
    }
}
```

2.2 **numPermutation.cpp**

```cpp
#include <iostream>

using namespace std;

void permutation(int A, int B, int C, int D, int result[24][4], int *counter) {
    int save[1000][4] = {};
    int count = 0;
    int col = 0;
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            for (int k = 0; k < 4; k++) {
                for (int l = 0; l < 4; l++) {
                    if (i != j && i != k && i != l && j != k && j != l && k != l) {
                        if (i == 0) {
                            save[count][col] = A;
                            col++;
                        }
                        if (i == 1) {
                            save[count][col] = B;
                            col++;
                        }
                        if (i == 2) {
                            save[count][col] = C;
                            col++;
                        }
                        if (i == 3) {
                            save[count][col] = D;
                            col++;
                        }
                        if (j == 0) {
                            save[count][col] = A;
                            col++;
                        }
                        if (j == 1) {
                            save[count][col] = B;
                            col++;
                        }
                        if (j == 2) {
                            save[count][col] = C;
                            col++;
                        }
                        if (j == 3) {
                            save[count][col] = D;
                            col++;
                        }
                        if (k == 0) {
                            save[count][col] = A;
```

```
                        col++;
                    }
                    if (k == 1) {
                        save[count][col] = B;
                        col++;
                    }
                    if (k == 2) {
                        save[count][col] = C;
                        col++;
                    }
                    if (k == 3) {
                        save[count][col] = D;
                        col++;
                    }
                    if (l == 0) {
                        save[count][col] = A;
                        col = 0;
                    }
                    if (l == 1) {
                        save[count][col] = B;
                        col = 0;
                    }
                    if (l == 2) {
                        save[count][col] = C;
                        col = 0;
                    }
                    if (l == 3) {
                        save[count][col] = D;
                        col = 0;
                    }
                    count++;
                }
            }
        }
    }
    int resultCount = 1;
    int resultCounts = 0;
    int saveCount = 1;
    result[0][0] = save[0][0];
    result[0][1] = save[0][1];
    result[0][2] = save[0][2];
    result[0][3] = save[0][3];
    while (saveCount < count) {
        for (int x = 0; x < resultCount; x++) {
            if (result[x][0] == save[saveCount][0] && result[x][1] == save[saveCount][1] &&
result[x][2] == save[saveCount][2] && result[x][3] == save[saveCount][3]) {
                break;
```

```
            }
            else {
                resultCounts++;
            }
            if (resultCounts == resultCount) {
                result[resultCount][0] = save[saveCount][0];
                result[resultCount][1] = save[saveCount][1];
                result[resultCount][2] = save[saveCount][2];
                result[resultCount][3] = save[saveCount][3];
                resultCount++;
            }
        }
        resultCounts = 0;
        saveCount++;
    }
    *counter = resultCount;
}
```

2.3 **operators.cpp**

```
#include <iostream>

using namespace std;

double operators(double a, double b, int n) {
    double result;
    if (n==1) {
        result = a + b;
    }
    else if (n==2) {
        result = a - b;
    }
    else if (n==3) {
        result = a * b;
    }
    else if (n==4) {
        result = a / b;
    }
    return result;
}

void opOutput(int i, int j, int k, string *op1, string *op2, string *op3) {
    if (i == 1) {
        *op1 = "+";
    }
    else if (i == 2) {
        *op1 = "-";
    }
    else if (i == 3) {
```

```
        *op1 = "*";
    }
    else if (i == 4) {
        *op1 = "/";
    }
    if (j == 1) {
        *op2 = "+";
    }
    else if (j == 2) {
        *op2 = "-";
    }
    else if (j == 3) {
        *op2 = "*";
    }
    else if (j == 4) {
        *op2 = "/";
    }
    if (k == 1) {
        *op3 = "+";
    }
    else if (k == 2) {
        *op3 = "-";
    }
    else if (k == 3) {
        *op3 = "*";
    }
    else if (k == 4) {
        *op3 = "/";
    }
}
```

2.4 **inputCard.cpp**

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

void inputTerminal(int *CardInt) {
    string Card[4];
    bool flag = false;
    cout << endl;
    cout << "Masukkan 4 kartu dengan input seperti berikut: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q,
K" << endl;
    while (flag == false) {
        cout << "Masukkan kartu: ";
        cin >> Card[0] >> Card[1] >> Card[2] >> Card[3];
        for (int i = 0; i < 4; i++) {
```

```
            if (Card[i] == "A" || Card[i] == "2" || Card[i] == "3" || Card[i] == "4" || Card[i] == "5" || Card[i] == "6" || Card[i] == "7" || Card[i] == "8" || Card[i] == "9" || Card[i] == "10" || Card[i] == "J" || Card[i] == "Q" || Card[i] == "K") {
                flag = true;
            }
            else {
                flag = false;
                cout << "!MASUKAN TIDAK SESUAI, MOHON ULANGI MASUKAN!" << endl;
                cout << endl;
                break;
            }
        }
    }
    for (int i = 0; i < 4; i++) {
        if (Card[i] == "A") {
            CardInt[i] = 1;
        }
        else if (Card[i] == "2") {
            CardInt[i] = 2;
        }
        else if (Card[i] == "3") {
            CardInt[i] = 3;
        }
        else if (Card[i] == "4") {
            CardInt[i] = 4;
        }
        else if (Card[i] == "5") {
            CardInt[i] = 5;
        }
        else if (Card[i] == "6") {
            CardInt[i] = 6;
        }
        else if (Card[i] == "7") {
            CardInt[i] = 7;
        }
        else if (Card[i] == "8") {
            CardInt[i] = 8;
        }
        else if (Card[i] == "9") {
            CardInt[i] = 9;
        }
        else if (Card[i] == "10") {
            CardInt[i] = 10;
        }
        else if (Card[i] == "J") {
            CardInt[i] = 11;
        }
        else if (Card[i] == "Q") {
```

```
            CardInt[i] = 12;
        }
        else if (Card[i] == "K") {
            CardInt[i] = 13;
        }
    }
}

void inputRandom(int *CardInt) {
    srand(time(NULL));
    CardInt[0] = (rand() % 13) + 1;
    CardInt[1] = (rand() % 13) + 1;
    CardInt[2] = (rand() % 13) + 1;
    CardInt[3] = (rand() % 13) + 1;
    cout << endl;
    cout << "Kartu yang didapat adalah: ";
    for (int i = 0; i < 4; i++) {
        if (CardInt[i] == 1) {
            cout << "A ";
        }
        else if (CardInt[i] == 11) {
            cout << "J ";
        }
        else if (CardInt[i] == 12) {
            cout << "Q ";
        }
        else if (CardInt[i] == 13) {
            cout << "K ";
        }
        else {
            cout << CardInt[i] << " ";
        }
    }
    cout << endl;
}
```

2.5 **saveDat.cpp**

```
#include <iostream>
#include <fstream>
#include <cstdlib>

using namespace std;

void save(string saveDat[], int count, int Cards[]) {
    ofstream outdata;
    int i;
    string input;
    bool flag = false;
```

```
    while (flag == false) {
        cout << "Apakah Anda ingin menyimpan file? (y/n): ";
        cin >> input;
        if (input == "y") {
            string fileName;
            cout << "Masukkan nama file: ";
            cin >> fileName;
            outdata.open("test/"+fileName+".txt");
            outdata << "KARTU: ";
            for (int i = 0; i < 4; i++) {
                if (Cards[i] == 1) {
                    outdata << "A ";
                }
                else if (Cards[i] == 2) {
                    outdata << "2 ";
                }
                else if (Cards[i] == 3) {
                    outdata << "3 ";
                }
                else if (Cards[i] == 4) {
                    outdata << "4 ";
                }
                else if (Cards[i] == 5) {
                    outdata << "5 ";
                }
                else if (Cards[i] == 6) {
                    outdata << "6 ";
                }
                else if (Cards[i] == 7) {
                    outdata << "7 ";
                }
                else if (Cards[i] == 8) {
                    outdata << "8 ";
                }
                else if (Cards[i] == 9) {
                    outdata << "9 ";
                }
                else if (Cards[i] == 10) {
                    outdata << "10 ";
                }
                else if (Cards[i] == 11) {
                    outdata << "J ";
                }
                else if (Cards[i] == 12) {
                    outdata << "Q ";
                }
                else if (Cards[i] == 13) {
                    outdata << "K ";
```

```
                }
            }
            if (count != 0) {
                outdata << endl;
                outdata << "====================" << endl;
                outdata << count << " SOLUSI DITEMUKAN!"<< endl;
                outdata << "====================" << endl;
                outdata << "Solusi:" << endl;
                for (i = 0; i < count; ++i)
                    outdata << saveDat[i] << endl;
                outdata << "====================" << endl;
            }
            else {
                outdata << endl;
                outdata << "=================" << endl;
                outdata << "TIDAK ADA SOLUSI!" << endl;
                outdata << "=================" << endl;
            }
            outdata.close();
            cout << "File berhasil disimpan!" << endl;
            flag = true;
        }
        else if (input == "n") {
            flag = true;
        }
        else {
            cout << "!MASUKAN TIDAK SESUAI, MOHON ULANGI MASUKAN!" << endl;
            cout << endl;
            flag = false;
        }
    }
}
```

2.6 **splashScreen.cpp**

```
#include <iostream>
#include <fstream>
#include <cstdlib>

using namespace std;

void welcomeScreen() {
    string myText;
    ifstream MyReadFile("src/others/splashScreen.txt");
    while (getline (MyReadFile, myText)) {
    cout << myText << endl;
    }
    MyReadFile.close();
}
```

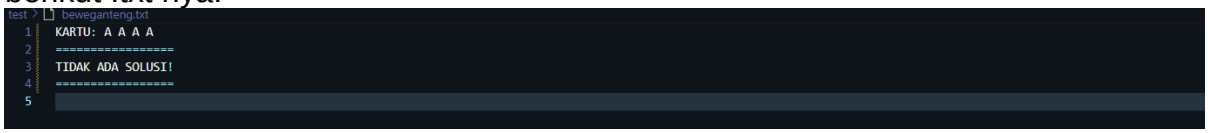
```
void menu() {
    cout << endl;
    cout << "Menu: " << endl;
    cout << "   1.  Input Manual" << endl;
    cout << "   2.  Input Random" << endl;
    cout << "   3.  Help" << endl;
    cout << "   4.  Exit" << endl;
    cout << "input: ";
}

void help() {
    cout << endl;
    cout << "Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan ";
    cout << "hasil akhir sejumlah 24. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan berhitung serta mengasah ";
    cout << "otak agar dapat berpikir dengan cepat dan akurat. Permainan Kartu 24 biasa dimainkan dengan menggunakan kartu remi. Kartu remi terdiri ";
    cout << "dari 52 kartu yang terbagi menjadi empat suit (sekop, hati, keriting, dan wajik) yang masing-masing terdiri dari 13 kartu (As, 2, 3, 4, ";
    cout << "5, 6, 7, 8, 9, 10, Jack, Queen, dan King). Yang perlu diperhatikan hanyalah nilai kartu yang didapat (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, ";
    cout << "Jack, Queen, dan King). As bernilai 1, Jack bernilai 11, Queen bernilai 12, King bernilai 13, sedangkan kartu bilangan memiliki nilai ";
    cout << "dari bilangan itu sendiri. Pada awal permainan moderator atau salah satu pemain mengambil 4 kartu dari dek yang sudah dikocok secara ";
    cout << "random. Permainan berakhir ketika pemain berhasil menemukan solusi untuk membuat kumpulan nilainya menjadi 24. Pengubahan nilai tersebut ";
    cout << "dapat dilakukan menggunakan operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian (X), divisi (/) dan tanda kurung ( () ). ";
    cout << "Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas." << endl;
    cout << endl;
    cout << "<> Contoh masukan untuk Menu nomor 1: " << endl;
    cout << "A 2 3 4 " << endl;
    cout << "10 J Q K " << endl;
    cout << endl;
    cout << "<> Menu nomor 2 hanya mengeluarkan kartu-kartu random." << endl;
    cout << endl;
    cout << "<> Setelah mendapatkan Solusi dan Jumlah Solusinya, Anda dapat menyimpan data tersebut ke dalam file .txt." << endl;
    cout << endl;
    cout << "Selamat Mencoba :D" << endl;
    system("pause");
}
```

3. Input dan Output

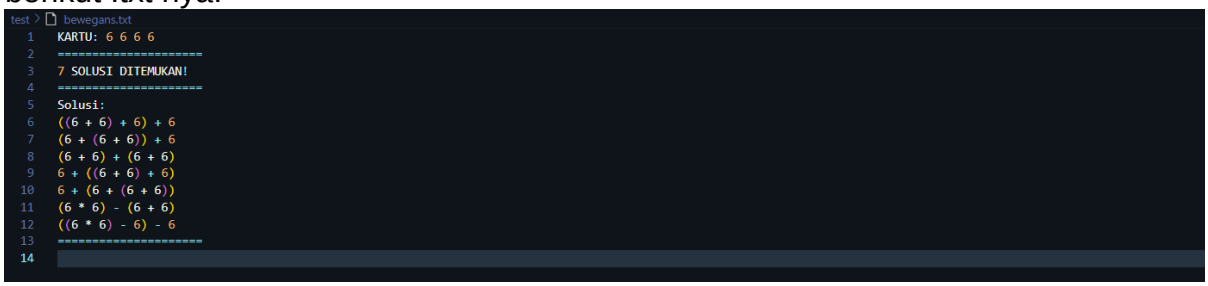
3.1 Contoh input manual tidak ada solusi (Input A A A A)

berikut .txt nya:



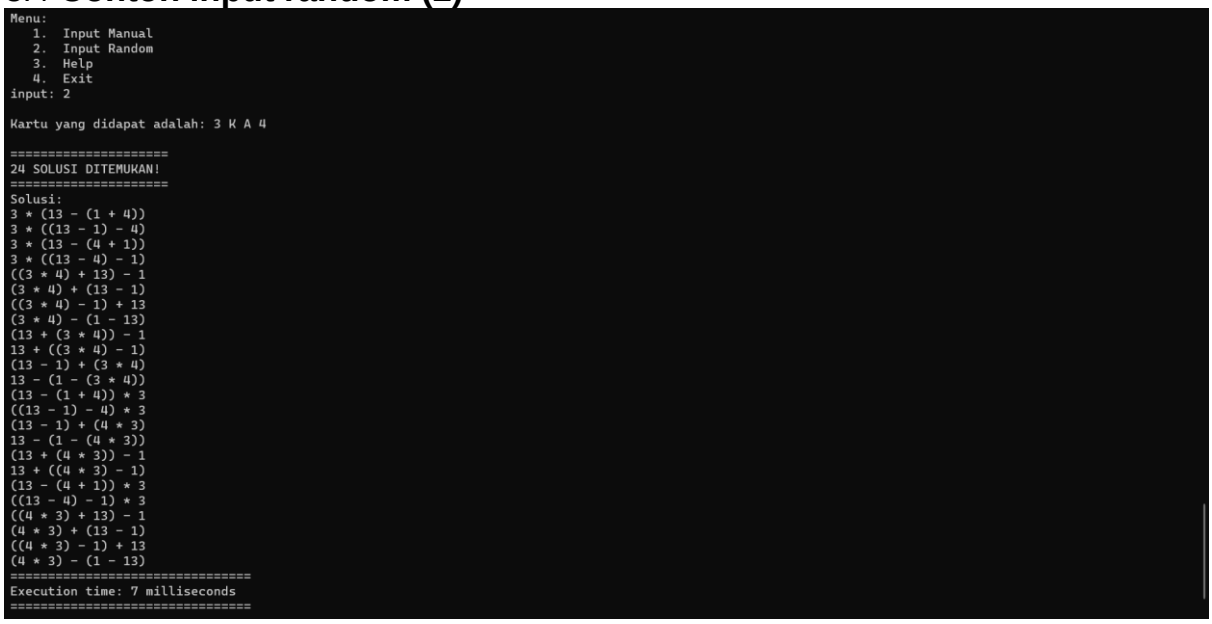
3.2 Contoh input manual ada solusi (Input 6 6 6 6)

berikut .txt nya:



3.3 **Contoh input random (1)**

3.4 **Contoh input random (2)**



3.5 **Contoh input random (3)**

3.6 **Contoh input random (4)**

4. Link Repo

https://github.com/bernarduswillson/Tucil1_13521021

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	v	
2. Program berhasil *running*	v	
3. Program dapat membaca input / generate sendiri dan memberikan luaran	v	
4. Solusi yang diberikan program memenuhi (berhasil mencapai 24)	v	
5. Program dapat menyimpan solusi dalam file teks	v	