IF2211 Strategi Algoritma

Laporan Tugas Kecil 1

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Oleh

Tabitha Permalla – 13521111

PROGRAM STUDI INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

1. Pendahuluan: Permainan Kartu 24

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan berhitung serta mengasah otak agar dapat berpikir dengan cepat dan akurat. Permainan Kartu 24 biasa dimainkan dengan menggunakan kartu remi. Kartu remi terdiri dari 52 kartu yang terbagi menjadi empat suit (sekop, hati, keriting, dan wajik) yang masing-masing terdiri dari 13 kartu (As, 2, 3, 4, 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, Jack, Queen, dan King). As bernilai 1, Jack bernilai 11, Queen bernilai 12, King bernilai 13, sedangkan kartu bilangan memiliki nilai dari bilangan itu sendiri. Pada awal permainan moderator atau salah satu pemain mengambil 4 kartu dari dek yang sudah dikocok secara random. Permainan berakhir ketika pemain berhasil menemukan solusi untuk membuat kumpulan nilainya menjadi 24. Pengubahan nilai tersebut dapat dilakukan menggunakan operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian (×), divisi (/) dan tanda kurung (()). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas. (Paragraf di atas dikutip dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah2016/MakalahStima-2016-038.pdf).

2. Penyelesaian Permainan Kartu 24 Menggunakan Algoritma Brute-Force

Permainan kartu 24 dapat diselesaikan menggunakan algoritma brute-force sebagai berikut:

1. Dicari semua permutasi yang mungkin dari kartu-kartu yang terpilih.
2. Untuk setiap permutasi kartu, dicoba semua kombinasi persamaan yang mungkin.
3. Untuk setiap kombinasi persamaan yang mungkin, dilakukan perhitungan terhadap persamaan tersebut. Apabila hasil perhitungan adalah 24, maka persamaan tersebut adalah salah satu solusi yang mungkin. Apabila hasil perhitungan tidak sama dengan 24, maka persamaan tersebut bukanlah salah satu solusi.

3. Source Code Program

Dalam pengerjaan tugas kecil ini, saya memutuskan untuk menggunakan bahasa pemrograman C. Program dibagi menjadi beberapa file sebagai berikut:

Nama File	Keterangan
main.c	Berisi program utama serta fungsi permutasi
boolean.h	Definisi type Boolean
equation.h	Definisi type equation
equation.c	Implementasi type equation
queuelinked.h	Definisi type queue
queuelinked.c	Implementasi type queue
evaluator.c	Berisi fungsi res dan eval yang mengembalikan nilai suatu persamaan
util.c	Berisi fungsi random dan swap guna memperlengkapi program utama

Berikut adalah source code program untuk tiap-tiap file:

main.c

```c
/* File : main.c */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "string.h"
#include "boolean.h"
#include "queuelinked.c"
#include "equation.c"
#include "util.c"
#include "evaluator.c"

int permutations[24][4];
int pcount = 0;
char ten[2] = {'1','0'};

void permute(int * card, int l){
    if (l == 4){
        for (int i = 0; i < 4; i++){
            permutations[pcount][i] = card[i];
        }
        pcount++;
    } else {
        for (int i = l; i < 4; i++){
            int duplicate = 0;
            for (int j = l; j < i; j++){
                if (card[i] == card[j]){
                    duplicate = 1;
                }
            }
            if (!duplicate){
                swap(card + l, card + i);
                permute(card, l + 1);
                swap(card + l, card + i);
            }
        }
    }
}


int main(){
    srand(time(0));
    clock_t start, end;
    boolean valid = false;
    int choice;
    Queue q;
    CreateQueue(&q);
    while (!valid){
        printf("How would you like to input the cards?\n  1. Manually from the keyboard\n  2. Use the random generator\n\nInput choice number: ");
        scanf("%d", &choice);
        if (choice == 1 || choice == 2){
            valid = true;
        }
    }
    int * cards;
    switch (choice){
    case 1:
```

```
        char temp[2];
        int i = 0;
        boolean invalid = false;
        while (i < 4){
            scanf("%s", &temp);
            boolean found = false;
            for (int j = 0; j < 13; j++){
                if (j == 9){
                    if (strcmp(temp,ten) == 0){
                        cards[i] = 10;
                        found = true;
                        break;
                    }
                } else if (temp[0] == cardlist[j] && strlen(temp) == 1){
                    cards[i] = j + 1;
                    found = true;
                    break;
                }
            }
            i++;
            if (!found){
                invalid = true;
            }
            if (i == 4){
                if (invalid){
                    printf("Card Invalid Found! Please re-enter 4 cards!\n");
                    i = 0;
                    invalid = false;
                }
            }
        }
        break;
    case 2:
        cards = random();
        break;
    default:
        break;
    }

    char ops[] = {'+', '-', '*', '/'};

    start = clock();
    int count = 0;
    int m = 0;
    permute(cards, 0);
    while (m < pcount){
        for (int i = 0; i < 4; i++){
            for (int j = 0; j < 4; j++){
                for (int k = 0; k < 4; k++){
                    count = eval(&q,permutations[m], ops[i], ops[j], ops[k], count);
                }
            }
        }
        m++;
    }
    end = clock();

    printf("There are %d solution(s)\n", count);
    DisplayQueue(q);

    // Execution time
    double exec_time = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Execution time: %f seconds\n", exec_time);

    // Save to file
    int choice2;
    boolean valid2 = false;
    while (!valid2){
        printf("Would you like to safe to file?\n  1. Yes\n  2. No\n\nInput choice number: ");
        scanf("%d", &choice2);
        if (choice2 == 1 || choice2 == 2){
            valid2 = true;
        }
    }

    switch (choice2) {
    case 1:
        boolean fvalid = false;
        char file[50];
        while (!fvalid){
            printf("Name of file with extension (.txt): ");
            scanf("%s", &file);
            int l = strlen(file);
            if (l > 4 && file[l - 4] == '.' && file[l - 3] == 't' && file[l - 2] == 'x' && file[l - 1] == 't'){
                fvalid = true;
            }
        }
        char path[100] = "../test/";
        for (int i = 0; i < strlen(file); i++){
            path[8+i] = file[i];
        }
        FILE *fptr;
        fptr = fopen(path, "w");
        if (fptr == NULL){
            printf("Error!\n");
        } else {
            fprintf(fptr, "Cards: ");
            for (int i = 0; i < 4; i++){
                if (cards[i] == 10){
```

```
                    fprintf(fptr, "10 ");
                } else{
                    fprintf(fptr, "%c ", cardlist[cards[i] - 1]);
                }
            }
            fprintf(fptr, "\nThere are %d solution(s)\n", count);
            if (!isEmpty(q)){
                Address p = ADDR_HEAD(q);
                while (NEXT(p) != NULL){
                    writeEqToFile(fptr, p->Eq);
                    p = NEXT(p);
                }
                writeEqToFile(fptr, p->Eq);
            }
        }
        fclose(fptr);
        break;

    default:
        break;
    }
}
```

boolean.h

```
/* Definisi type boolean */

#ifndef _BOOLEAN_h
#define _BOOLEAN_h

#define boolean unsigned char
#define true 1
#define false 0

#endif
```

equation.h

```
/* File: equation.h */

#ifndef EQUATION_H
#define EQUATION_H

typedef struct {
    int type;
    int * cards;
    char c1;
    char c2;
    char c3;
} Equation;

#define type(E) (E).type
#define cards(E) (E).cards
#define c1(E) (E).c1
#define c2(E) (E).c2
#define c3(E) (E).c3

void setEquation(Equation * Eq, int T, int * set, char a, char b, char c);

void displayEquation(Equation Eq);

void writeEqToFile(FILE * f, Equation Eq);

#endif
```

equation.c

```c
/* File: equation.c */

#include <stdio.h>
#include "equation.h"

void setEquation(Equation * Eq, int T, int * set, char a, char b, char c){
    type(*Eq) = T;
    cards(*Eq) = set;
    c1(*Eq) = a;
    c2(*Eq) = b;
    c3(*Eq) = c;
}

void displayEquation(Equation Eq){
    switch (Eq.type) {
    case 1:
        printf("(%d %c %d) %c (%d %c %d) = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    case 2:
        printf("((%d %c %d) %c %d) %c %d = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    case 3:
        printf("(%d %c (%d %c %d)) %c %d = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    case 4:
        printf("%d %c ((%d %c %d) %c %d) = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    case 5:
        printf("%d %c (%d %c (%d %c %d)) = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    default:
        break;
    }
}

void writeEqToFile(FILE * f, Equation Eq){
    switch (Eq.type) {
    case 1:
        fprintf(f,"(%d %c %d) %c (%d %c %d) = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    case 2:
        fprintf(f,"((%d %c %d) %c %d) %c %d = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    case 3:
        fprintf(f,"(%d %c (%d %c %d)) %c %d = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    case 4:
        fprintf(f,"%d %c ((%d %c %d) %c %d) = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    case 5:
        fprintf(f,"%d %c (%d %c (%d %c %d)) = 24\n", Eq.cards[0], Eq.c1, Eq.cards[1], Eq.c2, Eq.cards[2], Eq.c3, Eq.cards[3]);
        break;

    default:
        break;
    }
}
```

queuelinked.h

```c
/* File: queuelinked.h */

#ifndef QUEUELINKED_H
#define QUEUELINKED_H
#include "boolean.h"
#include "equation.h"
#include <stdlib.h>

#define NIL NULL

/* Queue dengan representasi berkait dengan pointer */
typedef struct node* Address;
typedef struct node {
    Equation Eq;
    Address next;
} Node;
/* Type queue dengan ciri HEAD dan TAIL: */
typedef struct {
    Address addrHead; /* alamat penghapusan */
    Address addrTail; /* alamat penambahan */
} Queue;
```

```c
/* Selektor */
#define NEXT(p) (p)->next
#define Equation(p) (p)->Eq

#define ADDR_HEAD(q) (q).addrHead
#define ADDR_TAIL(q) (q).addrTail
#define      HEAD(q) (q).addrHead->info

/* Prototype manajemen memori */
Address newNode(Equation eq);
/* Mengembalikan alamat sebuah Node hasil alokasi dengan info = x,
   atau NIL jika alokasi gagal */
boolean isEmpty(Queue q);
/* Mengirim true jika q kosong: ADDR_HEAD(q)=NULL and ADDR_TAIL(q)=NULL */
int length(Queue q);
/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika q kosong */

/*** Kreator ***/
void CreateQueue(Queue *q);
/* I.S. sembarang */
/* F.S. Sebuah q kosong terbentuk */

/*** Primitif Enqueue/Dequeue ***/
void enqueue(Queue *q, Equation eq);
/* Proses: Mengalokasi x dan menambahkan x pada bagian Tail dari q
           jika alokasi berhasil; jika alokasi gagal q tetap */
/* Pada dasarnya adalah proses insertLast */
/* I.S. q mungkin kosong */
/* F.S. x menjadi Tail, Tail "maju" */

void DisplayQueue(Queue q);

#endif
```

queuelinked.c

```c
/* File: queuelinked.c */

#include <stdio.h>
#include "queuelinked.h"

/* Prototype manajemen memori */
Address newNode(Equation eq){
/* Mengembalikan alamat sebuah Node hasil alokasi dengan info = x,
   atau NIL jika alokasi gagal */
    Address p = (Address) malloc(sizeof(Node));
    if (p != NULL){
        Equation(p) = eq;
        NEXT(p) = NULL;
    }
    return p;
}

boolean isEmpty(Queue q){
/* Mengirim true jika q kosong: ADDR_HEAD(q)=NULL and ADDR_TAIL(q)=NULL */
    return ADDR_HEAD(q) == NULL && ADDR_TAIL(q) == NULL;
}

int length(Queue q){
/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika q kosong */
    Address p = ADDR_HEAD(q);
    int i = 0;
    while (p != NULL){
        i++;
        p = NEXT(p);
    }
    return i;
}

/*** Kreator ***/
void CreateQueue(Queue *q){
/* I.S. sembarang */
/* F.S. Sebuah q kosong terbentuk */
    ADDR_HEAD(*q) = NULL;
    ADDR_TAIL(*q) = NULL;
}

/*** Primitif Enqueue/Dequeue ***/
void enqueue(Queue *q, Equation eq){
/* Proses: Mengalokasi x dan menambahkan x pada bagian Tail dari q
           jika alokasi berhasil; jika alokasi gagal q tetap */
/* Pada dasarnya adalah proses insertLast */
/* I.S. q mungkin kosong */
/* F.S. x menjadi Tail, Tail "maju" */
    Address p = newNode(eq);
    if (p != NULL){
        if (isEmpty(*q)){
            ADDR_HEAD(*q) = p;
        } else{
            NEXT(ADDR_TAIL(*q)) = p;
        }
        ADDR_TAIL(*q) = p;
    }
}

void DisplayQueue(Queue q){
```

```c
/* Proses : Menuliskan isi Queue, ditulis di antara kurung siku; antara dua elemen
    dipisahkan dengan separator "koma", tanpa tambahan karakter di depan, di tengah,
    atau di belakang, termasuk spasi dan enter */
/* I.S. q boleh kosong */
/* F.S. Jika q tidak kosong: [e1,e2,...,en] */
/* Contoh : jika ada tiga elemen bernilai 1, 20, 30 akan dicetak: [1,20,30] */
/* Jika Queue kosong : menulis [] */
    if (!isEmpty(q)){
        Address p = ADDR_HEAD(q);
        while (NEXT(p) != NULL){
            displayEquation(p->Eq);
            p = NEXT(p);
        }
        displayEquation(p->Eq);
    }
}
```

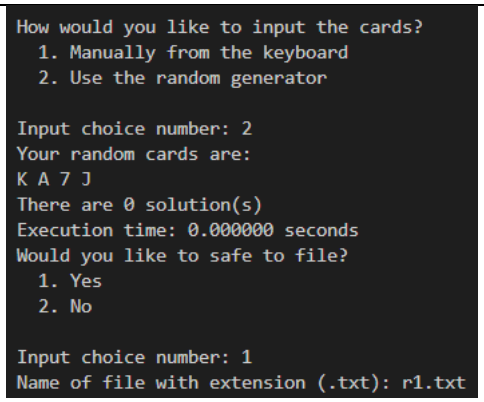
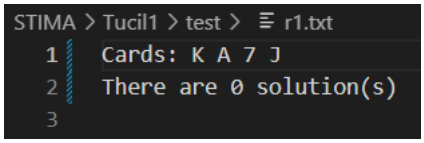
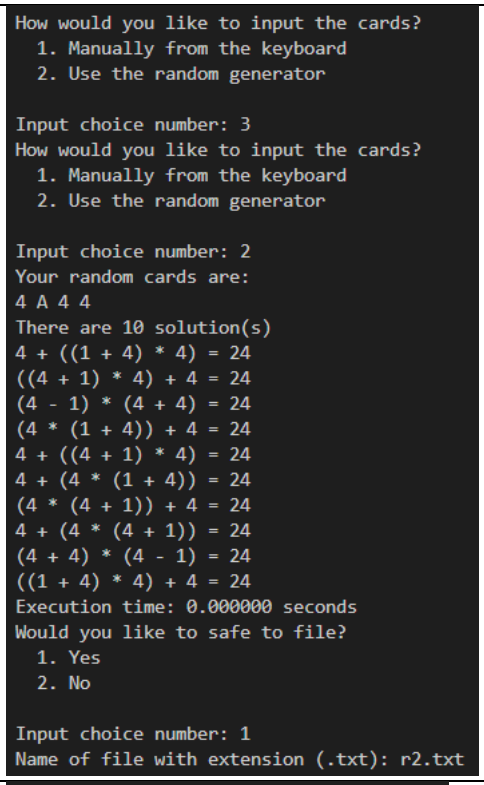
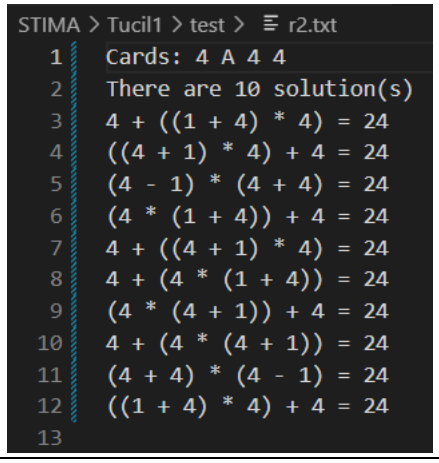
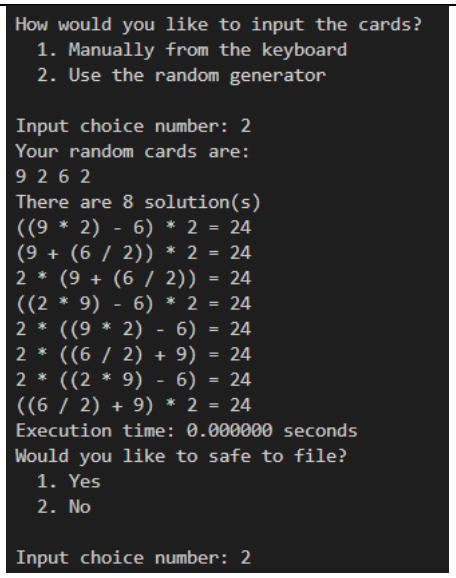
evaluator.c

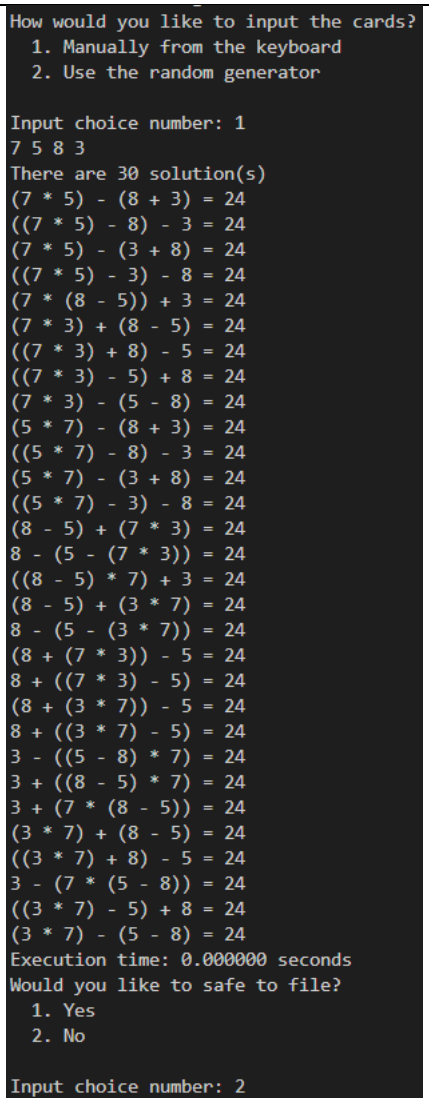
```c
/* File: evaluator.c */

#include "queuelinked.h"
#include "equation.h"

char ops[] = {'+', '-', '*', '/'};

float res(float a, float b, char c){
    if (c == '+'){
        return a + b;
    } else if (c == '-'){
        return a - b;
    } else if (c == '*'){
        return a * b;
    } else {
        return a / b;
    }
}

int eval(Queue * q, int * set, char c1, char c2, char c3, int count){
    float res1 = res(set[0], set[1], c1);   // a + b
    float res2 = res(set[1], set[2], c2);   // b + c
    float res3 = res(set[2], set[3], c3);   // c + d
    float res4 = res(res1, set[2], c2);     // (a + b) + c
    float res5 = res(set[0], res2, c1);     // a + (b + c)
    float res6 = res(res2, set[3], c3);     // (b + c) + d
    float res7 = res(set[1], res3, c2);     // b + (c + d)

    Equation eq;

    // (a + b) + (c + d)
    if (res(res1,res3,c2) == 24.00){
        setEquation(&eq, 1, set, c1, c2, c3);
        enqueue(q, eq);
        count++;
    }
    // ((a + b) + c) + d
    if (res(res4,set[3],c3) == 24.00){
        setEquation(&eq, 2, set, c1, c2, c3);
        enqueue(q, eq);
        count++;
    }
    // (a + (b + c)) + d
    if (res(res5,set[3],c3) == 24.00){
        setEquation(&eq, 3, set, c1, c2, c3);
        enqueue(q, eq);
        count++;
    }
    // a + ((b + c) + d)
    if (res(set[0],res6,c1) == 24.00){
        setEquation(&eq, 4, set, c1, c2, c3);
        enqueue(q, eq);
        count++;
    }
    // a + (b + (c + d))
    if (res(set[0],res7,c1) == 24.00){
        setEquation(&eq, 5, set, c1, c2, c3);
        enqueue(q, eq);
        count++;
    }
    return count;
}
```

util.c

```c
/* File: util.c */

#include <stdio.h>
#include <stdlib.h>

char cardlist[13] = {'A','2','3','4','5','6','7','8','9','X','J','Q','K'};

int * random(){
    static int card[4];
    printf("Your random cards are: \n");
    for (int i = 0; i < 4; i++){
        card[i] = rand() % 13 + 1;
        if (card[i] == 10){
            printf("10 ");
        } else{
            printf("%c ", cardlist[card[i] - 1]);
        }
    }
    printf("\n");
    return card;
}

void swap(int * a, int * b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

4. *Testing*

Test Case	Result
1	 
2	 
3	

4	 How would you like to input the cards? 1. Manually from the keyboard 2. Use the random generator Input choice number: 1 7 5 8 3 There are 30 solution(s) (7 * 5) - (8 + 3) = 24 ((7 * 5) - 8) - 3 = 24 (7 * 5) - (3 + 8) = 24 ((7 * 5) - 3) - 8 = 24 (7 * (8 - 5)) + 3 = 24 (7 * 3) + (8 - 5) = 24 ((7 * 3) + 8) - 5 = 24 ((7 * 3) - 5) + 8 = 24 (7 * 3) - (5 - 8) = 24 (5 * 7) - (8 + 3) = 24 ((5 * 7) - 8) - 3 = 24 (5 * 7) - (3 + 8) = 24 ((5 * 7) - 3) - 8 = 24 (8 - 5) + (7 * 3) = 24 8 - (5 - (7 * 3)) = 24 ((8 - 5) * 7) + 3 = 24 (8 - 5) + (3 * 7) = 24 8 - (5 - (3 * 7)) = 24 (8 + (7 * 3)) - 5 = 24 8 + ((7 * 3) - 5) = 24 (8 + (3 * 7)) - 5 = 24 8 + ((3 * 7) - 5) = 24 3 - ((5 - 8) * 7) = 24 3 + ((8 - 5) * 7) = 24 3 + (7 * (8 - 5)) = 24 (3 * 7) + (8 - 5) = 24 ((3 * 7) + 8) - 5 = 24 3 - (7 * (5 - 8)) = 24 ((3 * 7) - 5) + 8 = 24 (3 * 7) - (5 - 8) = 24 Execution time: 0.000000 seconds Would you like to safe to file? 1. Yes 2. No Input choice number: 2	

5

```
How would you like to input the cards?
  1. Manually from the keyboard
  2. Use the random generator

Input choice number: 1
a b c d
Card Invalid Found! Please re-enter 4 cards!
a 2 3 4
Card Invalid Found! Please re-enter 4 cards!
A 5 5 6
There are 12 solution(s)
((1 + 5) * 5) - 6 = 24
((5 + 1) * 5) - 6 = 24
(5 * (1 + 5)) - 6 = 24
(5 * (5 + 1)) - 6 = 24
(5 * 6) - (5 + 1) = 24
((5 * 6) - 5) - 1 = 24
(5 * 6) - (1 + 5) = 24
((5 * 6) - 1) - 5 = 24
(6 * 5) - (5 + 1) = 24
((6 * 5) - 5) - 1 = 24
(6 * 5) - (1 + 5) = 24
((6 * 5) - 1) - 5 = 24
Execution time: 0.000000 seconds
Would you like to safe to file?
  1. Yes
  2. No

Input choice number: 1
Name of file with extension (.txt): r5.txt
```

STIMA > Tucil1 > test > r5.txt

```
1   Cards: A 5 5 6
2   There are 12 solution(s)
3   ((1 + 5) * 5) - 6 = 24
4   ((5 + 1) * 5) - 6 = 24
5   (5 * (1 + 5)) - 6 = 24
6   (5 * (5 + 1)) - 6 = 24
7   (5 * 6) - (5 + 1) = 24
8   ((5 * 6) - 5) - 1 = 24
9   (5 * 6) - (1 + 5) = 24
10  ((5 * 6) - 1) - 5 = 24
11  (6 * 5) - (5 + 1) = 24
12  ((6 * 5) - 5) - 1 = 24
13  (6 * 5) - (1 + 5) = 24
14  ((6 * 5) - 1) - 5 = 24
15
```

6

```
How would you like to input the cards?
  1. Manually from the keyboard
  2. Use the random generator

Input choice number: 1
A 10 5 K
There are 8 solution(s)
(10 / 5) * (13 - 1) = 24
10 / (5 / (13 - 1)) = 24
(10 * (13 - 1)) / 5 = 24
10 * ((13 - 1) / 5) = 24
((13 - 1) / 5) * 10 = 24
(13 - 1) / (5 / 10) = 24
(13 - 1) * (10 / 5) = 24
((13 - 1) * 10) / 5 = 24
Execution time: 0.001000 seconds
Would you like to safe to file?
  1. Yes
  2. No

Input choice number: 1
Name of file with extension (.txt): r
Name of file with extension (.txt): r6
Name of file with extension (.txt): r6.tc
Name of file with extension (.txt): .txt
Name of file with extension (.txt): r6.txt
```

STIMA > Tucil1 > test > r6.txt

```
1   Cards: A 10 5 K
2   There are 8 solution(s)
3   (10 / 5) * (13 - 1) = 24
4   10 / (5 / (13 - 1)) = 24
5   (10 * (13 - 1)) / 5 = 24
6   10 * ((13 - 1) / 5) = 24
7   ((13 - 1) / 5) * 10 = 24
8   (13 - 1) / (5 / 10) = 24
9   (13 - 1) * (10 / 5) = 24
10  ((13 - 1) * 10) / 5 = 24
11
```

5. *Repository*

Berikut adalah tautan Repository dari project ini:

https://github.com/Bitha17/Tucil1_13521111

6. Checklist

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	✓	