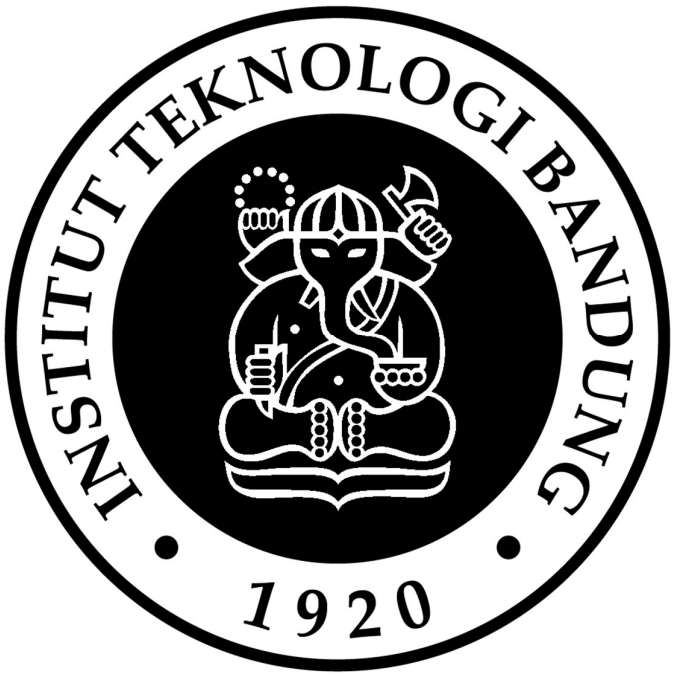
Laporan Tugas Kecil 1 IF2211 Strategi Algoritma Semester II Tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*



Disusun oleh:

Wilson Tansil 13521054

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Bandung

2023

Daftar Isi

Bab I
Deskripsi Masalah

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.

Spesifikasi program yang diharuskan:

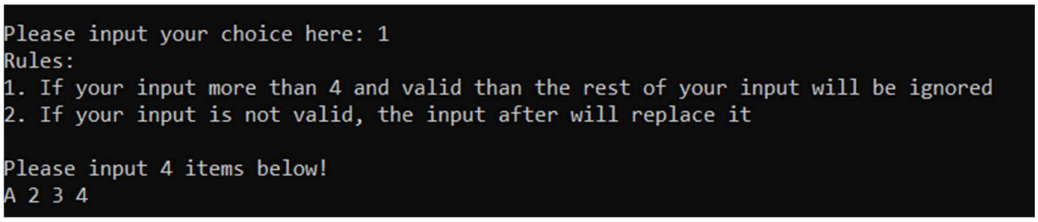
1. Program ditulis secara sederhana dalam Bahasa C/C++/Java yang mengimplementasikan algoritma Brute Force untuk mencari seluruh solusi permainan kartu 24.
2. Input: 4 angka/huruf yang terdiri dari: (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K).
   Contoh input: A 8 9 Q
   Selain itu, input juga dapat dilakukan dengan program men-generate 4 angka/hurufnya sendiri secara random. Pengguna dapat memilih apakah program meminta input dari pengguna atau generate sendiri. Apabila masukan tidak sesuai, maka program menampilkan luaran “Masukan tidak sesuai” dan akan meminta ulang.
3. Output:
   1. Banyaknya solusi yang ditemukan.
   2. Solusi dari permainan kartu 24 ditampilkan di layar dan terdapat opsi untuk menyimpan solusi dalam file text. Untuk contoh kasus di atas A 8 9 Q, maka salah satu solusinya adalah:
      ((9 + A) - 8) * Q atau ((9 + 1) - 8) * 12
      Note: Format penulisan output yang dicetak tidak harus persis contoh, yang penting merepresentasikan solusinya sudah cukup. Output apabila tidak ada solusi untuk pasangan kombinasi input, cukup ditampilkan “Tidak ada solusi”. Untuk solusi setiap masukan, perlu dipertimbangkan urutan nilai (x1..x4), urutan operator, dan grouping dengan kurung yang mungkin. Di akhir, program akan menanyakan “Apakah ingin menyimpan solusi ?”. Jika iya, program akan meminta sebuah nama untuk file teks dan semua solusi yang didapat akan disimpan dalam file text tersebut.
   3. Waktu eksekusi program (tidak termasuk waktu pembacaan file input).

Bab 2
Penjelasan Program dan Algoritma *Brute Force* yang Digunakan

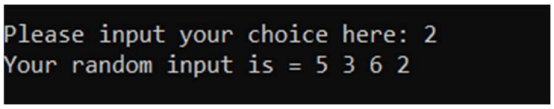
Algoritma Brute Force merupakan pendekatan yang straight forward untuk memecahkan suatu persoalan. Algoritma Brute Force memiliki keunggulan tersendiri yakni dapat memecahkan semua jenis persoalan sekaligus. Namun kekuarangannya adalah apabila berhadapan dengan jumlah data yang melewati batas memori atau memiliki kapasitas yang besar, maka device yang dijalankan pun harus memiliki memori dan processor yang cepat. Apabila tidak, maka algoritma ini tidak akan dapat dijalankan atau memakan waktu yang sangat lama. Dalam pengerjaan tugas kecil ini, bagian kode dibagi menjadi 3 bagian yakni bagian input, bagian process dan bagian output.

Bagian input merupakan bagian dimana terdapat baris yang mempertanyakan apakah ingin dirandom atau memasukan input sendiri. Apabila random maka, akan dilakukan pemilihan 4 kartu secara random. Jika memilih untuk melakukan input sendiri, maka akan disediakan tempat dan pengisian tersebut terdapat beberapa rule untuk melakukan input.

Gambar 2.1 Home page



Gambar 2.2 User Input

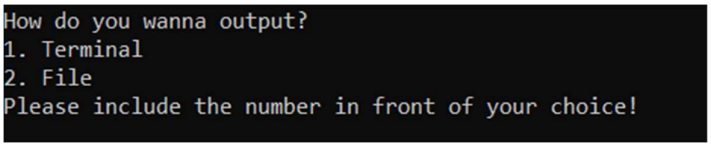


Gambar 2.3 Random Input

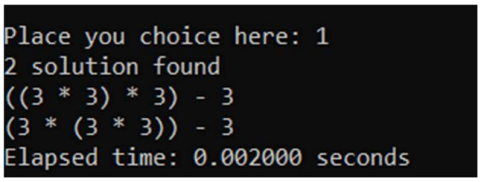
Selanjutnya adalah bagian proses, bagian ini adalah bagian yang menggunakan algoritma *brute force* sebagai algoritma utama penyelesaian masalah. Input yang didapatkan baik secara mandiri atau random akan diacak untuk mendapatkan semua permutasi dari keempat input tersebut. Dalam pencarian ini, digunakan operasi rekursif dan *swapping* untuk melakukannya. Setelah mendapatkan seluruh permutasi dan jumlahnya, maka akan disimpan

ke dalam global variable dengan tipe data yang telah ditentukan. Lalu langkah yang sama digunakan untuk pencarian semua permutasi yang mungkin dengan panjang 3 dari 4 operator, yakni +, -, / dan *.Setelah itu maka akan dilakukan penyesuaian kurung dengan mengkombinasikannya sesuai kombinasi kurung yang mungkin dapat dilakukan. Untuk kasus kali ini, terdapat 5 peletakkan kurung yang mungkin yakni ((___)__)__, (__(___))__, __((___)__), __(__(___)) dan (___)_(___). Setiap kemungkinan gabungan dari operator dan input akan dikombinasikan lagi dengan 5 kombinasi kurung ini dan apabila terdapat yang berjumlah 24 maka akan disimpan di global variable lain yang dikhususkan untuk result.

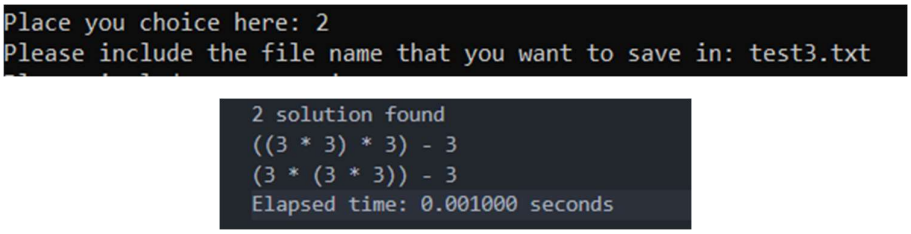
Terakhir, bagian output akan menanyakan terlebih dahulu apakah ingin diperlihatkan pada terminal atau pada file txt. Apabila memilih terminal maka akan langsung diperlihatkan apabila memilih file txt, maka jawaban tersebut akan disimpan pada file txt sesuai nama file yang dimasukkan pengguna. Juga terdapat bagian restart untuk program, apabila ingin menampilkannya dalam bentuk file maka harus mengakhiri programnya terlebih dahulu.



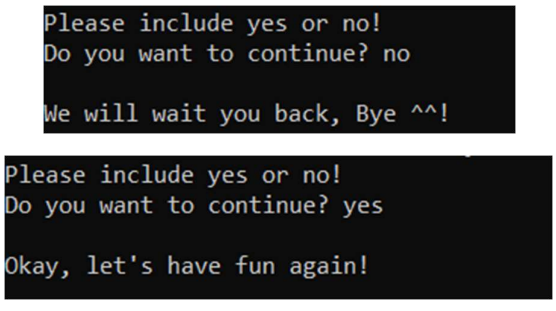
Gambar 2.4 Output Choice



Gambar 2.4 Output Using Terminal



Gambat 2.5 Output Using File



Gambar 2.6 Restart Choice

Bab 3
Kode Program Dalam Bahasa C

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

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

// input data type and decalaration
typedef struct{
    double array[4];
    int length;
}data;

data input_data;

// input data combination data type and declaration
typedef struct{
    double all_combination_data[1000][4];
    int length;
}combination_data;

combination_data combi_data;

// operator declaration
enum operator {PLUS = 1, MINUS = 2, DIVIDE = 3, MUL = 4};

int operator_arr[4] = {PLUS, MINUS, DIVIDE, MUL};

char operator_char[4] = {'+', '-', '/', '*'};

// operator combination data type and declaration
typedef struct{
    int all_combination_data[1000][3];
    int length;
}combination_op;

combination_op combi_op;

// result combination data type and declaration
typedef struct{
    char all_combination_data[1000][1000];
    int length;
}combination_res;

combination_res result_storage;

// stopwatch declaration
clock_t start_time, end_time;

// result of an expression between two number
double result_count(double first_num, int op, double second_num){
    if(op == 1){
        return first_num + second_num;
    }else if(op == 2){
        return first_num - second_num;
    }else if(op == 3){
        if(second_num != 0){
            return first_num / second_num;
        }else{
            return -28561; // if the divisor is 0, then it will be minus 13^4 (the greatest amount of the card possible) so that it is impossible to result 24
        }
    }else if(op == 4){
        return first_num * second_num;
    }
}

// find all operator combination (take 3 from 4)
void operator_permute(){
    for(int i= 0; i< 4; i++){
        for(int j= 0; j< 4; j++){
            for(int k= 0; k< 4; k++){
                combi_op.all_combination_data[combi_op.length][0] = operator_arr[i];
                combi_op.all_combination_data[combi_op.length][1] = operator_arr[j];
                combi_op.all_combination_data[combi_op.length][2] = operator_arr[k];
                combi_op.length++;
            }
        }
    }
}
```

```
// store the result in result data type that had been declared before
void convert_operation_and_store(double * num, int op[], int type){
    char str[100];
    if(type == 5){
        sprintf(str, "(%0.f %c %0.f) %c (%0.f %c %0.f)", num[0], operator_char[(op[0])-1], num[1], operator_char[(op[1])-1], num[2], operator_char[(op[2])-1], num[3]);
    }else if(type == 4){
        sprintf(str, "%0.f %c (%0.f %c (%0.f %c %0.f))", num[0], operator_char[(op[0])-1], num[1], operator_char[(op[1])-1], num[2], operator_char[(op[2])-1], num[3]);
    }else if(type == 3){
        sprintf(str, "%0.f %c ((%0.f %c %0.f) %c %0.f)", num[0], operator_char[(op[0])-1], num[1], operator_char[(op[1])-1], num[2], operator_char[(op[2])-1], num[3]);
    }else if(type == 2){
        sprintf(str, "(%0.f %c (%0.f %c %0.f)) %c %0.f", num[0], operator_char[(op[0])-1], num[1], operator_char[(op[1])-1], num[2], operator_char[(op[2])-1], num[3]);
    }else if(type == 1){
        sprintf(str, "((%0.f %c %0.f) %c %0.f) %c %0.f", num[0], operator_char[(op[0])-1], num[1], operator_char[(op[1])-1], num[2], operator_char[(op[2])-1], num[3]);
    }
    boolean duplicate = false;
    // find if there is duplicate value
    for(int i= 0; i< result_storage.length && !duplicate; i++){
        if(strcmp(result_storage.all_combination_data[i], str) == 0){
            duplicate = true;
        }
    }
    // if there is no duplicate value than it is valid to be stored
    if(!duplicate){
        strcpy(result_storage.all_combination_data[result_storage.length], str);
        result_storage.length++;
    }
}
```

```
// game progression place
void games_operator(){
    for(int i= 0; i< combi_op.length; i++){
        for(int j= 0; j< combi_data.length; j++){
            // every bracket combination
            double combination_bracket1 = result_count(result_count(result_count(combi_data.all_combination_data[j][0], combi_op.all_combination_data[i][0], combi_data.all_combination_data[j][1]), combi_op.all_combination_data[i][1], combi_data.all_combination_data[j][2]), combi_op.all_combination_data[i][2], combi_data.all_combination_data[j][3]);
            double combination_bracket2 = result_count(result_count(combi_data.all_combination_data[j][0], combi_op.all_combination_data[i][0], result_count(combi_data.all_combination_data[j][1], combi_op.all_combination_data[i][1], combi_data.all_combination_data[j][2])), combi_op.all_combination_data[i][2], combi_data.all_combination_data[j][3]);
            double combination_bracket3 = result_count(combi_data.all_combination_data[j][0], combi_op.all_combination_data[i][0], result_count(result_count(combi_data.all_combination_data[j][1], combi_op.all_combination_data[i][1], combi_data.all_combination_data[j][2]), combi_op.all_combination_data[i][2], combi_data.all_combination_data[j][3]));
            double combination_bracket4 = result_count(combi_data.all_combination_data[j][0], combi_op.all_combination_data[i][0], result_count(combi_data.all_combination_data[j][1], combi_op.all_combination_data[i][1], result_count(combi_data.all_combination_data[j][2], combi_op.all_combination_data[i][2], combi_data.all_combination_data[j][3])));
            double combination_bracket5 = result_count(result_count(combi_data.all_combination_data[j][0], combi_op.all_combination_data[i][0], combi_data.all_combination_data[j][1]), combi_op.all_combination_data[i][1], result_count(combi_data.all_combination_data[j][2], combi_op.all_combination_data[i][2], combi_data.all_combination_data[j][3]));
            // if the bracket combination result is 24
            if(combination_bracket1 == 24){
                convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 1);
            }
            if(combination_bracket2 == 24){
                convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 2);
            }
            if(combination_bracket3 == 24){
                convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 3);
            }
            if(combination_bracket4 == 24){
                convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 4);
            }
            if(combination_bracket5 == 24){
                convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 5);
            }
        }
    }
}
```

```
// swapping two double variable
void swap(double *a, double *b) {
    double temp = *a;
    *a = *b;
    *b = temp;
}

// find all combination of input data using recursion
void insert_combination_to_matrix(double list[], int start, int end) {
    if (start == end) {
        for (int i = 0; i <= end; i++) {
            combi_data.all_combination_data[combi_data.length][i] = list[i];
        }
        combi_data.length++;
    }
    else {
        for (int i = start; i <= end; i++) {
            swap(&list[start], &list[i]);
            insert_combination_to_matrix(list, start + 1, end);
            swap(&list[start], &list[i]);
        }
    }
}

// get combination of input data and operator
void get_all_combination(){
    insert_combination_to_matrix(input_data.array, 0, 3);
    operator_permute();
}

// the begining of the process where we find all the combination and process it
void brute_force(){
    start_time = clock();
    get_all_combination();
    games_operator();
}
```

```c
// input random data which is validated as instruction which is poker card
void input_random(){
    srand(time(NULL));
    printf("Your random input is = ");
    while(input_data.length < 4){
        input_data.array[input_data.length] = (double)(rand()%13)+1;
        if(input_data.array[input_data.length] == 13){
            printf("K ");
        }else if(input_data.array[input_data.length] == 12){
            printf("Q ");
        }
        else if(input_data.array[input_data.length] == 11){
            printf("J ");
        }else{
            printf("%0.f ", input_data.array[input_data.length]);
        }
        input_data.length++;
    }
    printf("\n");
}

```

```c
// read user input and convert to double
void read_input(){
    char input[5];
    double temp;
    for (int i = 0; i < 4; i++) {
        scanf("%s", input);
        if (strlen(input) == 1 && input[0] > '1' && input[0] <= '9') {
            temp = input[0] - '0';
        } else if (strlen(input) == 2 && input[0] == '1' && input[1] == '0') {
            temp = 10;
        } else if (strlen(input) == 1 && (input[0] == 'A' || input[0] == 'a')) {
            temp = 1;
        } else if (strlen(input) == 1 && (input[0] == 'K' || input[0] == 'k')) {
            temp = 13;
        } else if (strlen(input) == 1 && (input[0] == 'Q' || input[0] == 'q')) {
            temp = 12;
        } else if (strlen(input) == 1 && (input[0] == 'J' || input[0] == 'j')) {
            temp = 11;
        } else {
            printf("Invalid input, please enter a valid number\n");
            i--;
            continue;
        }
        input_data.array[input_data.length] = temp;
        input_data.length++;
    }
    // counter for input in one line
    fgets(input, 10000, stdin);
}

// user input beginning and instruction
void input_user(){
    printf(
        "Rules:\n"
        "1. If your input more than 4 and valid than the rest of your input will be ignored\n"
        "2. If your input is not valid, the input after will replace it\n\n"
        "Please input 4 items below!\n"
    );
    read_input();
}

```

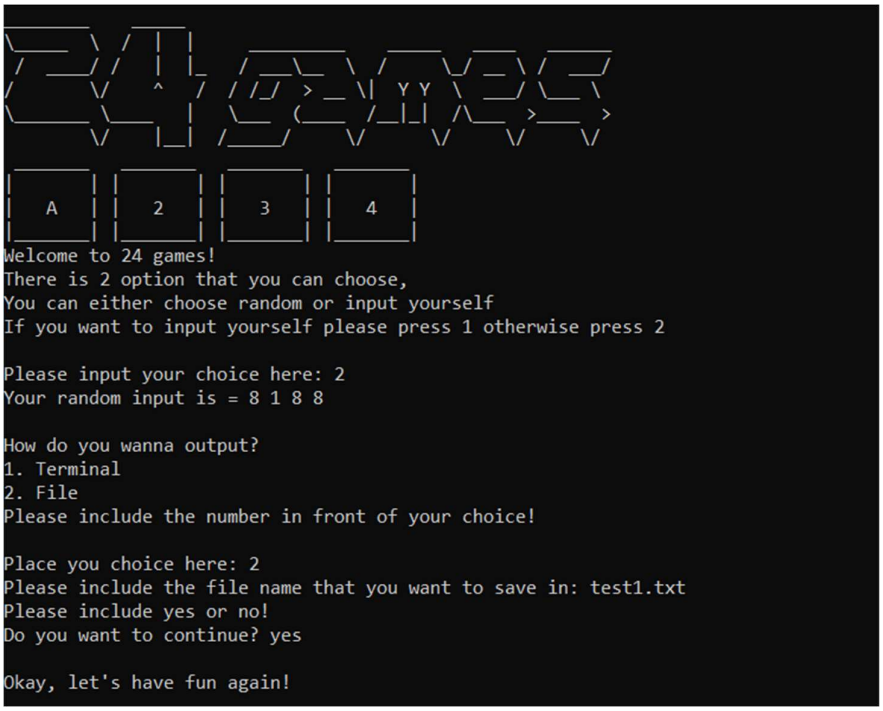
```c
// user input beginning and instruction
void input_user(){
    printf(
        "Rules:\n"
        "1. If your input more than 4 and valid than the rest of your input will be ignored\n"
        "2. If your input is not valid, the input after will replace it\n\n"
        "Please input 4 items below!\n"
    );
    read_input();
}

// user input choice with validator wheter user want to random input or input their own
void input_choice(){
    result_storage.length = 0;
    combi_data.length = 0;
    input_data.length = 0;
    int choice;
    printf("\nPlease input your choice here: ");
    scanf("%d", &choice);
    if(choice == 1){
        input_user();
    }else if(choice == 2){
        input_random();
    }else{
        printf("Your input is wrong please try again!\n");
        input_choice();
    }
}
```

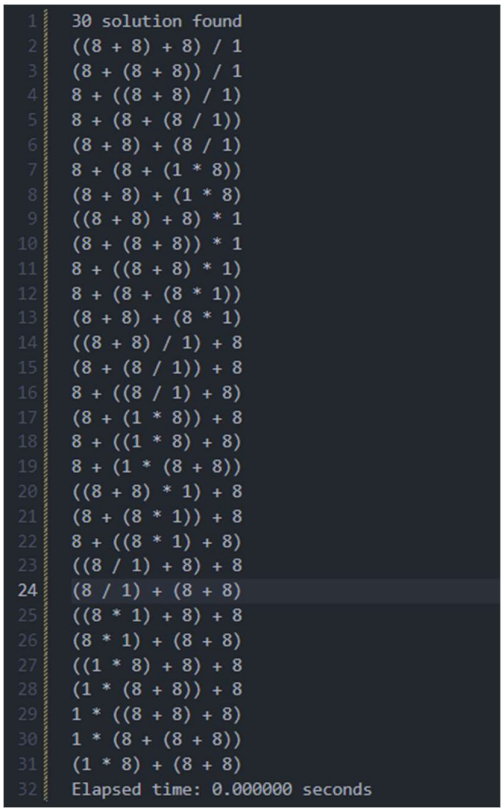
```c
// output part with some instruction wheter want to display on terminal or file and validator
void output_choice();

void progress_output(int choice){
    double total_time;
    total_time = (double)(end_time - start_time) / CLOCKS_PER_SEC;
    if(choice == 1){
        printf("%d solution found\n", result_storage.length);
        for(int i= 0; i< result_storage.length; i++){
            printf("%s\n", result_storage.all_combination_data[i]);
        }
        printf("Elapsed time: %f seconds\n", total_time);
    }else if(choice == 2){
        char file_name[100];
        printf("Please include the file name that you want to save in: ");
        scanf("%s", file_name);
        FILE * file;
        char path[200] = "../test/";
        strcat(path, file_name);
        file = fopen(path, "w");
        if(file == NULL){
            printf("Error opening file!\n");
        }else{
            fprintf(file, "%d solution found\n", result_storage.length);
            for(int i= 0; i< result_storage.length; i++){
            fprintf(file, "%s\n", result_storage.all_combination_data[i]);
            }
            fprintf(file, "Elapsed time: %f seconds", total_time);
        }
    }else{
        printf("Your input is wrong, Please try again!\n");
        output_choice();
    }
}
```

```c
void output_choice(){
    int choice_output;
    printf(
        "\nHow do you wanna output?\n"
        "1. Terminal\n"
        "2. File\n"
        "Please include the number in front of your choice!\n"
    );
    printf("\nPlace you choice here: ");
    scanf("%d", &choice_output);
    progress_output(choice_output);
}

// restart progress
void home();

void restart_choice();

void restart_progress(char * restart){
    if(strcmp(restart, "yes") == 0){
        printf("\nOkay, let's have fun again!\n");
        home();
    }else if(strcmp(restart, "no") == 0){
        printf("\nWe will wait you back, Bye ^^!\n");
        exit(0);
    }else{
        printf("\nYour input is wrong, Please reinput!\n");
        restart_choice();
    }
}

void restart_choice(){
    char restart[10];
    printf(
        "Please include yes or no!\n"
        "Do you want to continue? "
    );
    scanf("%s", restart);
    restart_progress(restart);
}
```

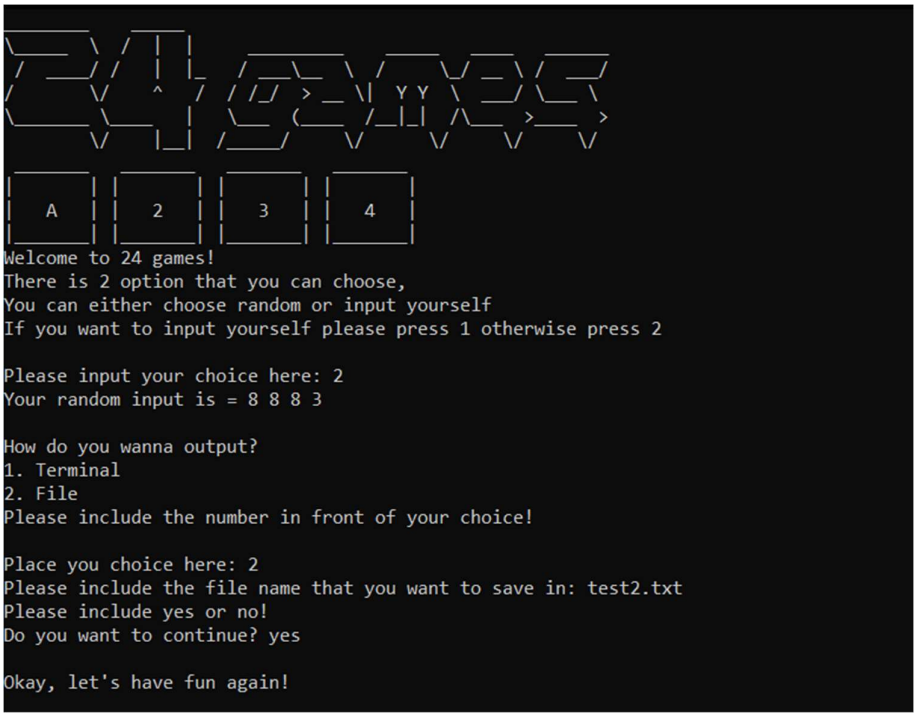
```c
// home page
void home(){
    printf(
"  ________    _____                                    \n"
"\\\_____  \\  /  |  |    ___________    _____   ____   ______\n"
" /  ____/ /   |  |_   / ___\\__  \\  /     \\_/ __ \\ /  ___/\n"
"/       \\/    ^   /  / /_/  > __ \\|  Y Y  \\  ___/ \\___ \\ \n"
"\\_______ \\____   |   \\___  (____  /__|_|  /\\___  >____  >\n"
"        \\/    |__|  /_____/     \\/      \\/     \\/     \\/ \n"
"  _______   _______   _______   _______\n"
"|       | |       | |       | |       |\n"
"|   A   | |   2   | |   3   | |   4   |\n"
"|_______| |_______| |_______| |_______|\n"
        "Welcome to 24 games!\n"
        "There is 2 option that you can choose,\n"
        "You can either choose random or input yourself\n"
        "If you want to input yourself please press 1 otherwise press 2\n"
    );
    input_choice();
    brute_force();
    end_time= clock();
    output_choice();
    restart_choice();
}

// main
int main(){
    system("cls");
    home();
}
```

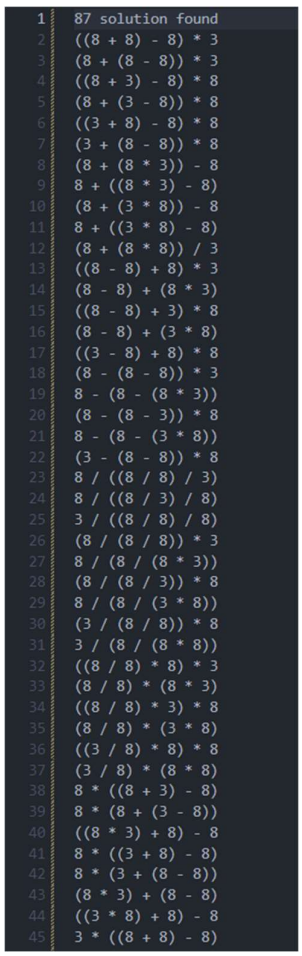
Bab 4
Input / Output Program

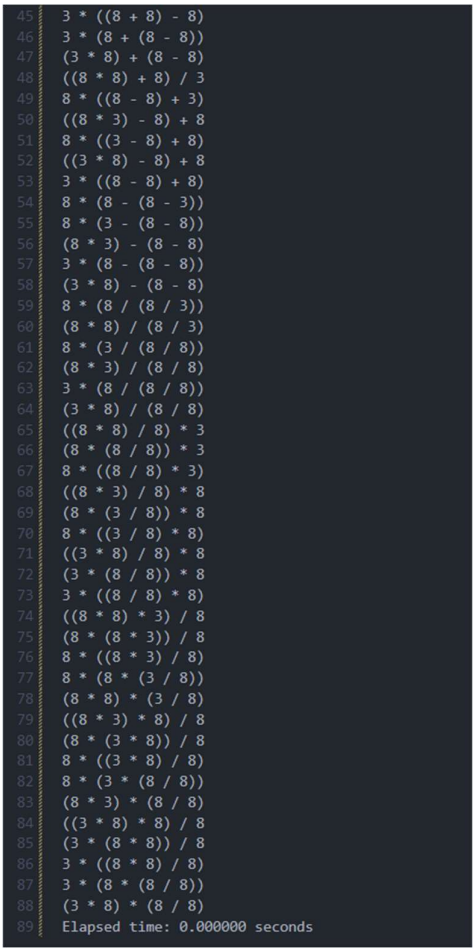




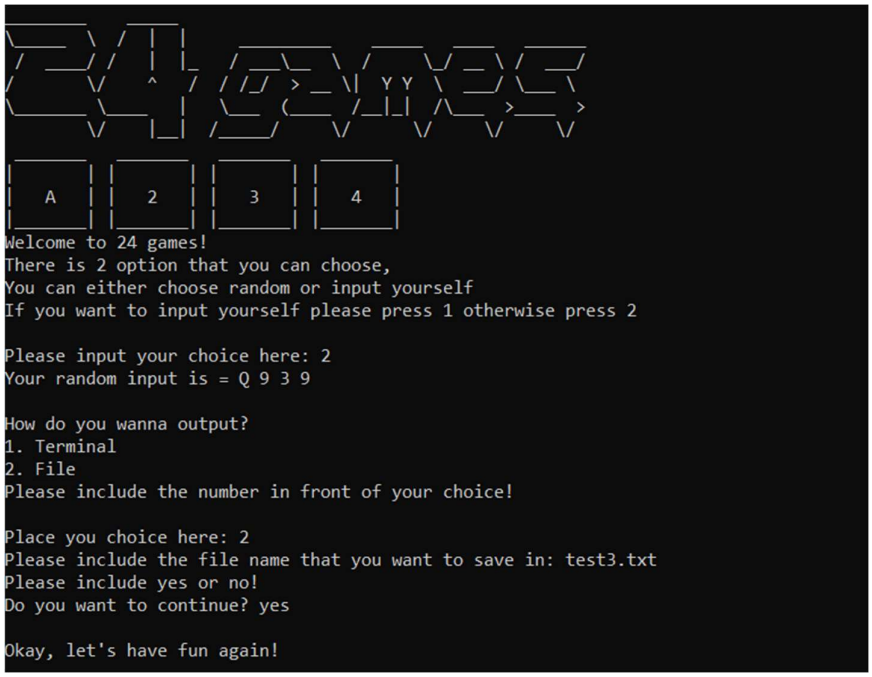
Gambar 4.1 Testcase Random 8 1 8 8

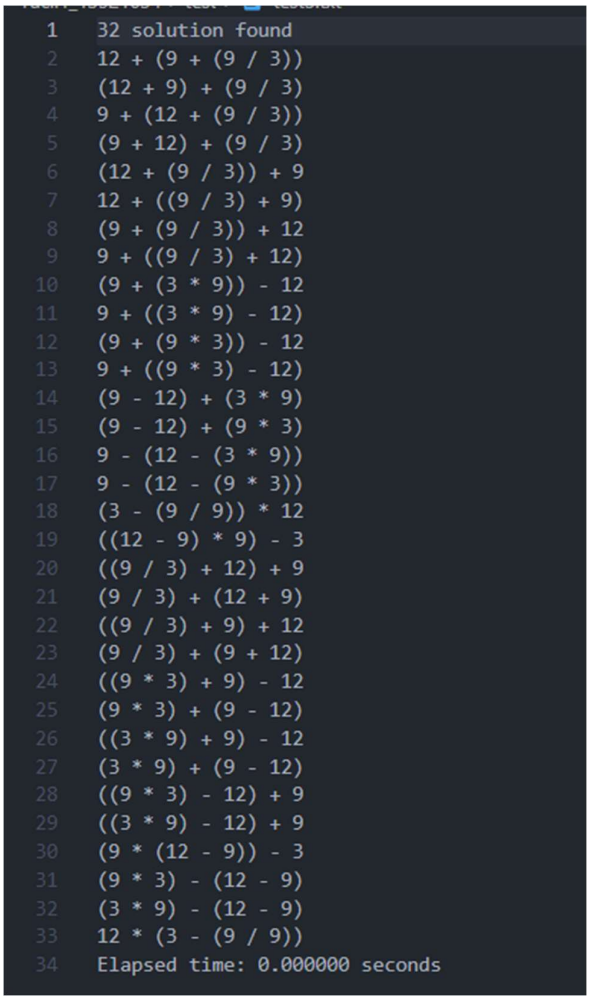






Gambar 4.2 Test Case Random 8 8 8 3

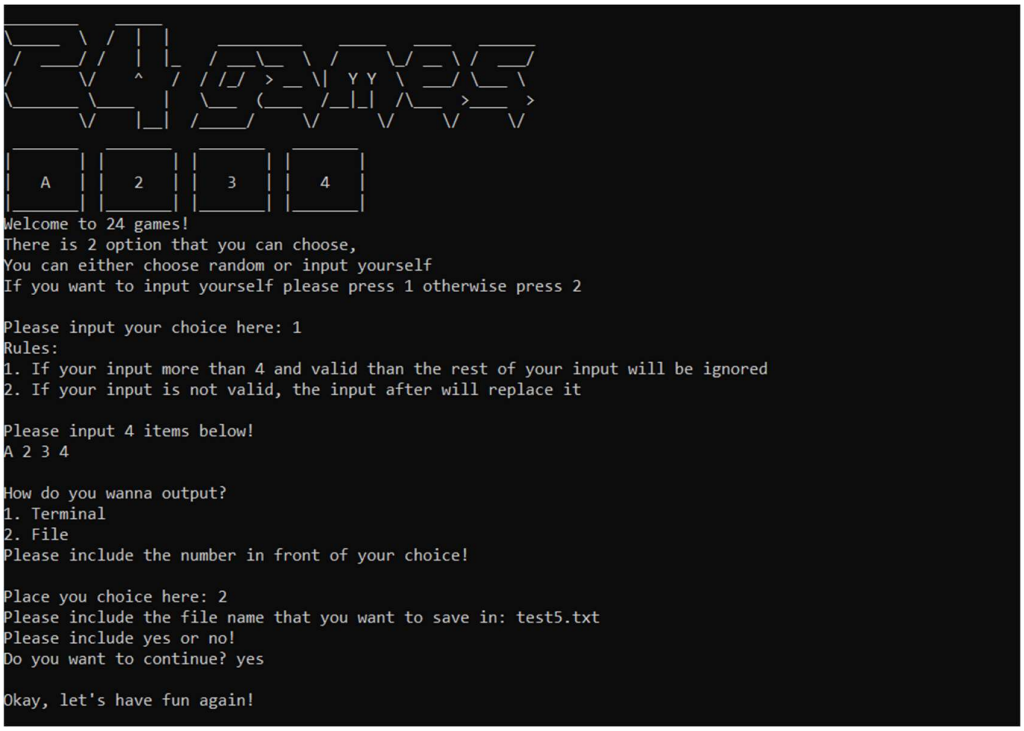




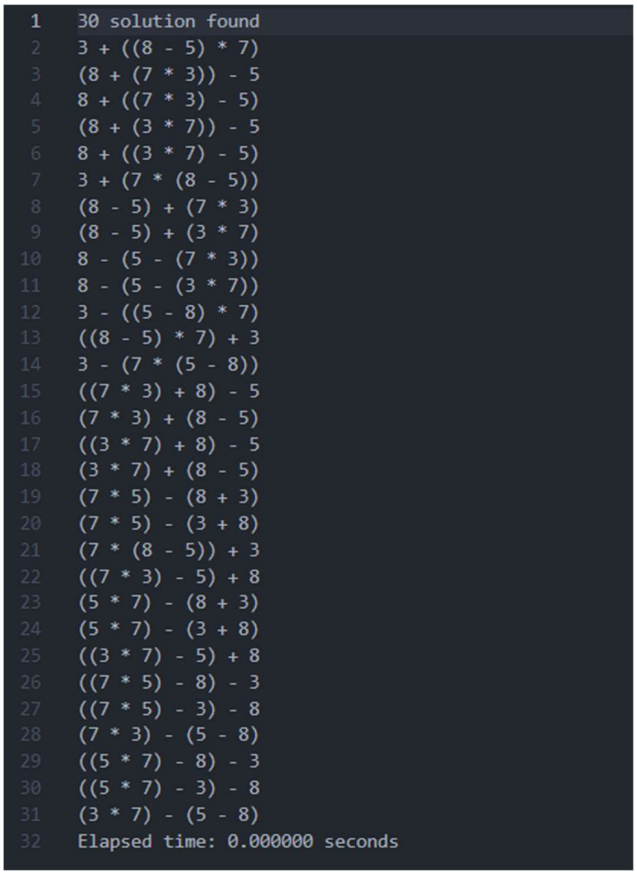
Gambar 4.3 Test Case Random Q 9 3 9


```
1   2 solution found
2   ((3 * 3) * 3) - 3
3   (3 * (3 * 3)) - 3
4   Elapsed time: 0.000000 seconds
```

Gambar 4.4 Test Case User Input 3 3 3 3



Gambar 4.5 Test Case User Input A 2 3 4



Gambar 4.6 Test Case User Input 7 5 8 3

Bab 5
Tabel Penilaian

Poin	Ya	Tidak
1. Program berhasil di kompilasi 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	✓	

Bab 6
Link Repository

Link : github