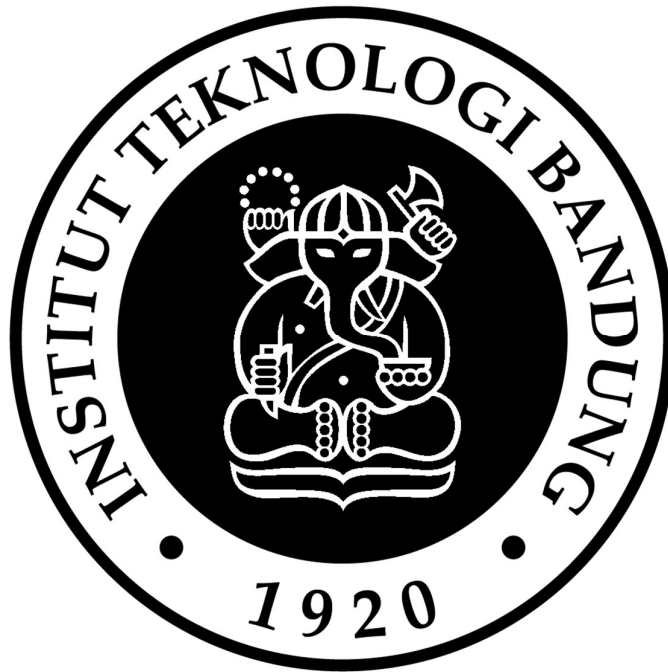


# **Laporan Tugas Kecil 1 IF2211 Strategi Algoritma Semester II Tahun 2022/2023**

**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 .....	3
Bab 2 Penjelasan Program dan Algoritma <i>Brute Force</i> yang digunakan.....	4
Bab 3 Kode Program dalam Bahasa C .....	5
Bab 4 Input / Output Program.....	10
Bab 5 Tabel Penilaian .....	16
Bab 6 Link Repository .....	16

# 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 ( $\times$ ), 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:  
$$((9 + A) - 8) * Q \text{ atau } ((9 + 1) - 8) * 12$$
  
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 kekurangannya 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

```
Please input your choice here: 1
Rules:
1. If your input more than 4 and valid than the rest of your input will be ignored
2. If your input is not valid, the input after will replace it

Please input 4 items below!
A 2 3 4
```

Gambar 2.2 User Input

```
Please input your choice here: 2
Your random input is = 5 3 6 2
```

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.

```
How do you wanna output?
1. Terminal
2. File
Please include the number in front of your choice!
```

*Gambar 2.4 Output Choice*

```
Place you choice here: 1
2 solution found
((3 * 3) * 3) - 3
(3 * (3 * 3)) - 3
Elapsed time: 0.002000 seconds
```

*Gambar 2.4 Output Using Terminal*

```
Place you choice here: 2
Please include the file name that you want to save in: test3.txt
```

```
2 solution found
((3 * 3) * 3) - 3
(3 * (3 * 3)) - 3
Elapsed time: 0.001000 seconds
```

*Gambat 2.5 Output Using File*

```
Please include yes or no!
Do you want to continue? no

We will wait you back, Bye ^^!
```

```
Please include yes or no!
Do you want to continue? yes

Okay, let's have fun again!
```

*Gambar 2.6 Restart Choice*

## Bab 3

### Kode Program Dalam Bahasa C

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  // boolean declaration
7  #define boolean unsigned char
8  #define true 1
9  #define false 0
10
11 // input data type and decalaration
12 typedef struct{
13     double array[4];
14     int length;
15 }data;
16
17 data input_data;
18
19 // input data combination data type and declaration
20 typedef struct{
21     double all_combination_data[1000][4];
22     int length;
23 }combination_data;
24
25 combination_data combi_data;
26
27 // operator declaration
28 enum operator {PLUS = 1, MINUS = 2, DIVIDE = 3, MUL = 4};
29
30 int operator_arr[4] = {PLUS, MINUS, DIVIDE, MUL};
31
32 char operator_char[4] = {'+', '-', '/', '**'};
33
34 // operator combination data type and declaration
35 typedef struct{
36     int all_combination_data[1000][3];
37     int length;
38 }combination_op;
39
40 combination_op combi_op;
41
```

```
42 // result combination data type and declaration
43 typedef struct{
44     char all_combination_data[1000][1000];
45     int length;
46 }combination_res;
47
48 combination_res result_storage;
49
50 // stopwatch declaration
51 clock_t start_time, end_time;
52
53 // result of an expression between two number
54 double result_count(double first_num, int op, double second_num){
55     if(op == 1){
56         return first_num + second_num;
57     }else if(op == 2){
58         return first_num - second_num;
59     }else if(op == 3){
60         if(second_num != 0){
61             return first_num / second_num;
62         }else{
63             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
64         }
65     }else if(op == 4){
66         return first_num * second_num;
67     }
68 }
69
70 // find all operator combination (take 3 from 4)
71 void operator_permute(){
72     for(int i= 0; i< 4; i++){
73         for(int j= 0; j< 4; j++){
74             for(int k= 0; k< 4; k++){
75                 combi_op.all_combination_data[combi_op.length][0] = operator_arr[i];
76                 combi_op.all_combination_data[combi_op.length][1] = operator_arr[j];
77                 combi_op.all_combination_data[combi_op.length][2] = operator_arr[k];
78                 combi_op.length++;
79             }
80         }
81     }
82 }
83
```

```

84 // store the result in result data type that had been declared before
85 void convert_operation_and_store(double * num, int op[], int type){
86     char str[100];
87     if(type == 5){
88         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]);
89     }else if(type == 4){
90         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]);
91     }else if(type == 3){
92         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]);
93     }else if(type == 2){
94         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]);
95     }else if(type == 1){
96         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]);
97     }
98     boolean duplicate = false;
99     // find if there is duplicate value
100     for(int i = 0; i < result_storage.length && !duplicate; i++){
101         if(strcmp(result_storage.all_combination_data[i], str) == 0){
102             duplicate = true;
103         }
104     }
105     // if there is no duplicate value than it is valid to be stored
106     if(!duplicate){
107         strcpy(result_storage.all_combination_data[result_storage.length], str);
108         result_storage.length++;
109     }
110 }
111

```

```

112 // game progression place
113 void games_operator(){
114     for(int i = 0; i < combi_op.length; i++){
115         for(int j = 0; j < combi_data.length; j++){
116             // every bracket combination
117             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]);
118             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]);
119             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]));
120             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])));
121             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]));
122             // if the bracket combination result is 24
123             if(combination_bracket1 == 24){
124                 convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 1);
125             }
126             if(combination_bracket2 == 24){
127                 convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 2);
128             }
129             if(combination_bracket3 == 24){
130                 convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 3);
131             }
132             if(combination_bracket4 == 24){
133                 convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 4);
134             }
135             if(combination_bracket5 == 24){
136                 convert_operation_and_store(combi_data.all_combination_data[j], combi_op.all_combination_data[i], 5);
137             }
138         }
139     }
140 }
141

```

```

142 // swapping two double variable
143 void swap(double *a, double *b) {
144     double temp = *a;
145     *a = *b;
146     *b = temp;
147 }
148
149 // find all combination of input data using recursion
150 void insert_combination_to_matrix(double list[], int start, int end) {
151     if (start == end) {
152         for (int i = 0; i <= end; i++) {
153             combi_data.all_combination_data[combi_data.length][i] = list[i];
154         }
155         combi_data.length++;
156     }
157     else {
158         for (int i = start; i <= end; i++) {
159             swap(&list[start], &list[i]);
160             insert_combination_to_matrix(list, start + 1, end);
161             swap(&list[start], &list[i]);
162         }
163     }
164 }
165
166 // get combination of input data and operator
167 void get_all_combination(){
168     insert_combination_to_matrix(input_data.array, 0, 3);
169     operator_permute();
170 }
171
172 // the beginning of the process where we find all the combination and process it
173 void brute_force(){
174     start_time = clock();
175     get_all_combination();
176     games_operator();
177 }
178

```

```

179 // input random data which is validated as instruction which is poker card
180 void input_random(){
181     srand(time(NULL));
182     printf("Your random input is = ");
183     while(input_data.length < 4){
184         input_data.array[input_data.length] = (double)(rand()%13)+1;
185         if(input_data.array[input_data.length] == 13){
186             printf("K ");
187         }else if(input_data.array[input_data.length] == 12){
188             printf("Q ");
189         }
190         else if(input_data.array[input_data.length] == 11){
191             printf("J ");
192         }else{
193             printf("%0.f ", input_data.array[input_data.length]);
194         }
195         input_data.length++;
196     }
197     printf("\n");
198 }
199

```

```

200 // read user input and convert to double
201 void read_input(){
202     char input[5];
203     double temp;
204     for (int i = 0; i < 4; i++) {
205         scanf("%s", input);
206         if (strlen(input) == 1 && input[0] > '1' && input[0] <= '9') {
207             temp = input[0] - '0';
208         } else if (strlen(input) == 2 && input[0] == '1' && input[1] == '0') {
209             temp = 10;
210         } else if (strlen(input) == 1 && (input[0] == 'A' || input[0] == 'a')) {
211             temp = 1;
212         } else if (strlen(input) == 1 && (input[0] == 'K' || input[0] == 'k')) {
213             temp = 13;
214         } else if (strlen(input) == 1 && (input[0] == 'Q' || input[0] == 'q')) {
215             temp = 12;
216         } else if (strlen(input) == 1 && (input[0] == 'J' || input[0] == 'j')) {
217             temp = 11;
218         } else {
219             printf("Invalid input, please enter a valid number\n");
220             i--;
221             continue;
222         }
223         input_data.array[input_data.length] = temp;
224         input_data.length++;
225     }
226     // counter for input in one line
227     fgets(input, 10000, stdin);
228 }
229
230 // user input beginning and instruction
231 void input_user(){
232     printf(
233         "Rules:\n"
234         "1. If your input more than 4 and valid than the rest of your input will be ignored\n"
235         "2. If your input is not valid, the input after will replace it\n\n"
236         "Please input 4 items below\n"
237     );
238     read_input();
239 }
240

```

```

240 // user input beginning and instruction
241 void input_user(){
242     printf(
243         "Rules:\n"
244         "1. If your input more than 4 and valid than the rest of your input will be ignored\n"
245         "2. If your input is not valid, the input after will replace it\n\n"
246         "Please input 4 items below\n"
247     );
248     read_input();
249 }
250
251 // user input choice with validator wheter user want to random input or input their own
252 void input_choice(){
253     result_storage.length = 0;
254     combi_data.length = 0;
255     input_data.length = 0;
256     int choice;
257     printf("\nPlease input your choice here: ");
258     scanf("%d", &choice);
259     if(choice == 1){
260         input_user();
261     }else if(choice == 2){
262         input_random();
263     }else{
264         printf("Your input is wrong please try again!\n");
265         input_choice();
266     }
267 }

```



```

259 // output part with some instruction whether want to display on terminal or file and validator
260 void output_choice();
261
262 void progress_output(int choice){
263     double total_time;
264     total_time = (double)(end_time - start_time) / CLOCKS_PER_SEC;
265     if(choice == 1){
266         printf("%d solution found\n", result_storage.length);
267         for(int i= 0; i< result_storage.length; i++){
268             printf("%s\n", result_storage.all_combination_data[i]);
269         }
270         printf("Elapsed time: %f seconds\n", total_time);
271     }else if(choice == 2){
272         char file_name[100];
273         printf("Please include the file name that you want to save in: ");
274         scanf("%s", file_name);
275         FILE * file;
276         char path[200] = "../test/";
277         strcat(path, file_name);
278         file = fopen(path, "w");
279         if(file == NULL){
280             printf("Error opening file!\n");
281         }else{
282             fprintf(file, "%d solution found\n", result_storage.length);
283             for(int i= 0; i< result_storage.length; i++){
284                 fprintf(file, "%s\n", result_storage.all_combination_data[i]);
285             }
286             fprintf(file, "Elapsed time: %f seconds", total_time);
287         }
288     }else{
289         printf("Your input is wrong, Please try again!\n");
290         output_choice();
291     }
292 }
293

```

```

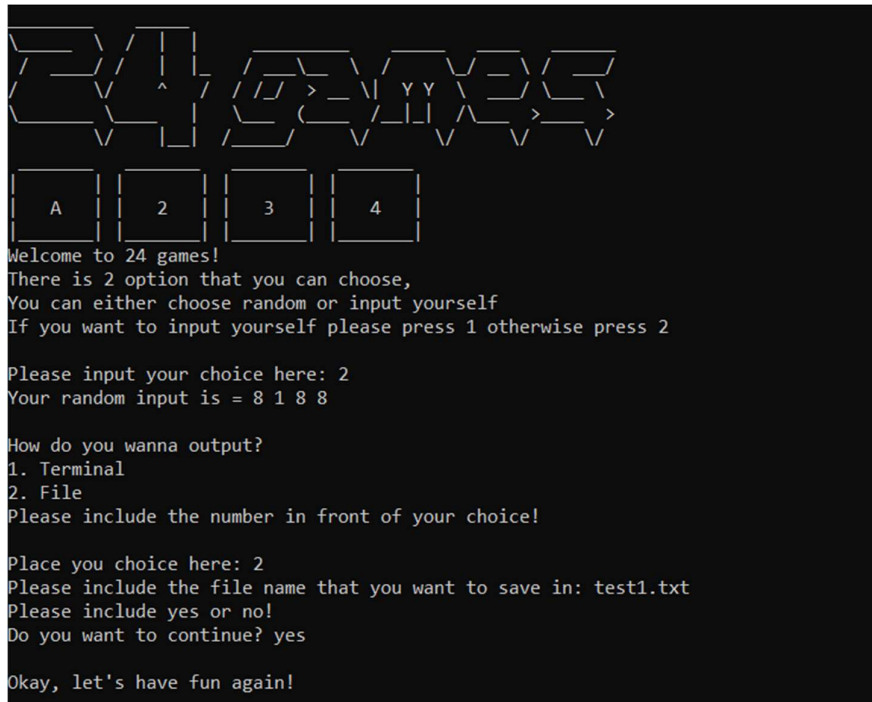
294 void output_choice(){
295     int choice_output;
296     printf(
297         "\nHow do you wanna output?\n"
298         "1. Terminal\n"
299         "2. File\n"
300         "Please include the number in front of your choice!\n"
301     );
302     printf("\nPlace you choice here: ");
303     scanf("%d", &choice_output);
304     progress_output(choice_output);
305 }
306
307 // restart progress
308 void home();
309
310 void restart_choice();
311
312 void restart_progress(char * restart){
313     if(strcmp(restart, "yes") == 0){
314         printf("\nOkay, let's have fun again!\n");
315         home();
316     }else if(strcmp(restart, "no") == 0){
317         printf("\nWe will wait you back, Bye ^^!\n");
318         exit(0);
319     }else{
320         printf("\nYour input is wrong, Please reinput!\n");
321         restart_choice();
322     }
323 }
324
325 void restart_choice(){
326     char restart[10];
327     printf(
328         "Please include yes or no!\n"
329         "Do you want to continue? "
330     );
331     scanf("%s", restart);
332     restart_progress(restart);
333 }
334

```

[illegible]

## Bab 4

### Input / Output Program



```
1 30 solution found
2 ((8 + 8) + 8) / 1
3 (8 + (8 + 8)) / 1
4 8 + ((8 + 8) / 1)
5 8 + (8 + (8 / 1))
6 (8 + 8) + (8 / 1)
7 8 + (8 + (1 * 8))
8 (8 + 8) + (1 * 8)
9 ((8 + 8) + 8) * 1
10 (8 + (8 + 8)) * 1
11 8 + ((8 + 8) * 1)
12 8 + (8 + (8 * 1))
13 (8 + 8) + (8 * 1)
14 ((8 + 8) / 1) + 8
15 (8 + (8 / 1)) + 8
16 8 + ((8 / 1) + 8)
17 (8 + (1 * 8)) + 8
18 8 + ((1 * 8) + 8)
19 8 + (1 * (8 + 8))
20 ((8 + 8) * 1) + 8
21 (8 + (8 * 1)) + 8
22 8 + ((8 * 1) + 8)
23 ((8 / 1) + 8) + 8
24 (8 / 1) + (8 + 8)
25 ((8 * 1) + 8) + 8
26 (8 * 1) + (8 + 8)
27 ((1 * 8) + 8) + 8
28 (1 * (8 + 8)) + 8
29 1 * ((8 + 8) + 8)
30 1 * (8 + (8 + 8))
31 (1 * 8) + (8 + 8)
32 Elapsed time: 0.000000 seconds
```

Gambar 4.1 Testcase Random 8 1 8 8





```

  24 GAMES
  _____
  | A | | 2 | | 3 | | 4 |
  |___|_|_|_|_|_|_|_|
Welcome to 24 games!
There is 2 option that you can choose,
You can either choose random or input yourself
If you want to input yourself please press 1 otherwise press 2

Please input your choice here: 1
Rules:
1. If your input more than 4 and valid than the rest of your input will be ignored
2. If your input is not valid, the input after will replace it

Please input 4 items below!
3 3 3 3

How do you wanna output?
1. Terminal
2. File
Please include the number in front of your choice!

Place you choice here: 2
Please include the file name that you want to save in: test4.txt
Please include yes or no!
Do you want to continue? yes

Okay, let's have fun again!

```

```

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



```

  _____
 /         \
/   A   \   /   2   \   /   3   \   /   4   \
 \         /
  _____

Welcome to 24 games!
There is 2 option that you can choose,
You can either choose random or input yourself
If you want to input yourself please press 1 otherwise press 2

Please input your choice here: 1
Rules:
1. If your input more than 4 and valid than the rest of your input will be ignored
2. If your input is not valid, the input after will replace it

Please input 4 items below!
A 2 3 4

How do you wanna output?
1. Terminal
2. File
Please include the number in front of your choice!

Place you choice here: 2
Please include the file name that you want to save in: test5.txt
Please include yes or no!
Do you want to continue? yes

Okay, let's have fun again!

```

1	242 solution found	96	$((3 / 1) * 4) * 2$	191	$((4 * 3) / 1) * 2$	386	$1 * ((3 * 2) * 4)$	581	$2 * ((4 * 1) * 3)$	776	$4 * ((3 * 2) * 1)$
2	$((1 + 2) + 3) * 4$	97	$(3 / 1) * (4 * 2)$	192	$4 * ((3 / 1)) * 2$	387	$1 * (3 * (2 * 4))$	582	$2 * (4 * (1 * 3))$	777	$4 * (3 * (2 * 1))$
3	$(1 + (2 + 3)) * 4$	98	$((4 / 1) * 3) * 2$	193	$4 * ((3 / 1) * 2)$	388	$(1 * 3) * (2 * 4)$	583	$(2 * 4) * (1 * 3)$	778	$(4 * 3) * (2 * 1)$
4	$((1 + 3) + 2) * 4$	99	$(4 / 1) * (3 * 2)$	194	$((2 * 3) * 4) / 1$	389	$((1 * 3) * 4) * 2$	584	$((3 * 2) * 1) * 4$	779	$((4 * 3) * 1) * 2$
5	$(1 + (3 + 2)) * 4$	100	$((4 / 1) * 2) * 3$	195	$(2 * (3 * 4)) / 1$	390	$(1 * (3 * 4)) * 2$	585	$(3 * (2 * 1)) * 4$	780	$(4 * (3 * 1)) * 2$
6	$((2 + 1) + 3) * 4$	101	$4 * ((3 * 2) + 1)$	196	$2 * ((3 * 4) / 1)$	391	$1 * ((3 * 4) * 2)$	586	$3 * ((2 * 1) * 4)$	781	$4 * ((3 * 1) * 2)$
7	$(2 + (1 + 3)) * 4$	102	$4 * ((2 + 3) + 1)$	197	$2 * (3 * (4 / 1))$	392	$(1 * 3) * (4 * 2)$	587	$3 * (2 * (1 * 4))$	782	$4 * (3 * (1 * 2))$
8	$((2 + 3) + 1) * 4$	103	$4 * (2 + (3 + 1))$	198	$(2 * 3) * (4 / 1)$	393	$(1 * 3) * (4 * 2)$	588	$(3 * 2) * (1 * 4)$	783	$(4 * 3) * (1 * 2)$
9	$(2 + (3 + 1)) * 4$	104	$4 * ((2 + 1) + 3)$	199	$((2 * 4) * 3) / 1$	394	$((1 * 4) * 3) * 2$	589	$(3 * (2 * 4)) * 1$	784	$((4 * 1) * 3) * 2$
10	$((3 + 2) + 1) * 4$	105	$4 * ((2 + 1 + 3))$	200	$(2 * (4 * 3)) / 1$	395	$(1 * (4 * 3)) * 2$	590	$(3 * (2 * 4)) * 1$	785	$(4 * (1 * 3)) * 2$
11	$(3 + (2 + 1)) * 4$	106	$4 * ((3 * 2) + 1)$	201	$2 * ((4 * 3) / 1)$	396	$1 * ((4 * 3) * 2)$	591	$3 * ((2 * 4) * 1)$	786	$4 * ((1 * 3) * 2)$
12	$((3 + 1) + 2) * 4$	107	$4 * (3 + (2 + 1))$	202	$2 * (4 * (3 / 1))$	397	$1 * (4 * (3 * 2))$	592	$3 * (2 * (4 * 1))$	787	$4 * (1 * (3 * 2))$
13	$(3 + (1 + 2)) * 4$	108	$4 * ((3 + 1) + 2)$	203	$(2 * 4) * (3 / 1)$	398	$(1 * 4) * (3 * 2)$	593	$(3 * 2) * (4 * 1)$	788	$(4 * 1) * (3 * 2)$
14	$(1 + 3) * (2 + 4)$	109	$4 * ((3 + 1 + 2))$	204	$((3 * 2) * 4) / 1$	399	$((1 * 4) * 2) * 3$	594	$((3 * 1) * 2) * 4$	789	$((4 * 1) * 2) * 3$
15	$(1 + 3) * (4 + 2)$	110	$4 * ((1 + 3) + 2)$	205	$(3 * (2 * 4)) / 1$	400	$(1 * (4 * 2)) * 3$	595	$(3 * (1 * 2)) * 4$	790	$(4 * (1 * 2)) * 3$
16	$(2 + 4) * (3 + 1)$	111	$4 * ((1 + 3 + 2))$	106	$3 * ((2 * 4) / 1)$	401	$1 * ((4 * 2) * 3)$	596	$3 * ((1 * 2) * 4)$	791	$4 * ((1 * 2) * 3)$
17	$(2 + 4) * (1 + 3)$	112	$4 * ((1 + 2) + 3)$	107	$3 * (2 * (4 / 1))$	402	$1 * (4 * (2 * 3))$	597	$3 * (1 * (2 * 4))$	792	$4 * (1 * (2 * 3))$
18	$(3 + 1) * (2 + 4)$	113	$4 * ((1 + 2 + 3))$	108	$(3 * 2) * (4 / 1)$	403	$(1 * 4) * (2 * 3)$	598	$(3 * 1) * (2 * 4)$	793	$4 * (1 * (2 * 3))$
19	$(3 + 1) * (4 + 2)$	114	$4 * ((3 * 4) * 2) / 1$	109	$((3 * 4) * 2) / 1$	404	$((2 * 1) * 3) * 4$	599	$((3 * 1) * 4) * 2$	794	$4 * (1 * (2 * 3))$
20	$(4 + 2) * (3 + 1)$	115	$(3 * (4 * 2)) / 1$	110	$(3 * (4 * 2)) / 1$	405	$(2 * (1 * 3)) * 4$	600	$(3 * (1 * 4)) * 2$	795	$4 * (1 * (2 * 3))$
21	$(4 + 2) * (1 + 3)$	116	$3 * ((4 * 2) / 1)$	111	$3 * (4 * (2 / 1))$	406	$2 * ((1 * 3) * 4)$	601	$3 * ((1 * 4) * 2)$	796	$4 * ((1 * 2) * 3)$
22	$2 / ((1 / 3) / 4)$	112	$3 * (4 * (2 / 1))$	112	$3 * (4 * (2 / 1))$	407	$2 * (1 * (3 * 4))$	602	$3 * (1 * (4 * 2))$	797	$4 * ((1 * 2) * 3)$
23	$2 / ((1 / 4) / 3)$	113	$(2 * 4) * (1 / 3)$	113	$(2 * 4) * (1 / 3)$	408	$(2 * 1) * (3 * 4)$	603	$(3 * 1) * (4 * 2)$	798	$4 * ((1 * 2) * 3)$
24	$3 / ((1 / 2) / 4)$	114	$3 * (2 / (1 / 4))$	114	$(4 * 2) * 3 / 1$	409	$(2 * 1) * (3 * 4)$	604	$(3 * 1) * (4 * 2)$	799	$4 * ((1 * 2) * 3)$
25	$3 / ((1 / 4) / 2)$	115	$(3 * 2) * (1 / 4)$	115	$(4 * 2) * 3 / 1$	410	$(2 * 1) * (3 * 4)$	605	$(3 * 1) * (4 * 2)$	800	$4 * ((1 * 2) * 3)$
26	$4 / ((1 / 3) / 2)$	116	$3 * (4 / (1 / 2))$	116	$4 * ((2 * 3) / 1)$	411	$2 * ((1 * 4) * 3)$	606	$3 * ((4 * 1) * 2)$	801	$4 * ((1 * 2) * 3)$
27	$4 / ((1 / 2) / 3)$	117	$4 * (2 * (1 / 3))$	117	$4 * ((2 * 3) / 1)$	412	$2 * (1 * (4 * 3))$	607	$3 * ((4 * 1) * 2)$	802	$4 * ((1 * 2) * 3)$
28	$(2 / (1 / 3)) * 4$	118	$(4 * 2) * (1 / 3)$	118	$(4 * 2) * (3 / 1)$	413	$(2 * 1) * (4 * 3)$	608	$(3 * 4) * (1 * 2)$	803	$(3 * 4) * (1 * 2)$
29	$2 / (1 / (3 * 4))$	119	$4 * (3 / (1 / 2))$	119	$((4 * 3) * 2) / 1$	414	$((2 * 3) * 1) * 4$	609	$(3 * 4) * (1 * 2)$	804	$(3 * 4) * (1 * 2)$
30	$(2 / (1 / 4)) * 3$	120	$(4 * 3) * (1 / 2)$	120	$(4 * (3 * 2)) / 1$	415	$(2 * (3 * 1)) * 4$	610	$(3 * (4 * 2)) * 1$	805	$(4 * (2 * 1)) * 3$
31	$2 / (1 / (4 * 3))$	121	$(2 * 3) * (1 * 4)$	121	$4 * ((3 * 2) / 1)$	416	$2 * ((3 * 1) * 4)$	611	$3 * ((4 * 2) * 1)$	806	$4 * ((2 * 1) * 3)$
32	$(3 / (1 / 2)) * 4$	122	$(2 * (3 / 1)) * 4$	122	$4 * (3 * (2 / 1))$	417	$2 * (3 * (1 * 4))$	612	$3 * (4 * (2 * 1))$	807	$4 * ((2 * 1) * 3)$
33	$3 / (1 / (2 * 4))$	123	$2 * ((3 / 1) * 4)$	123	$(4 * 3) * (2 / 1)$	418	$(2 * 3) * (1 * 4)$	613	$(3 * 4) * (2 * 1)$	808	$4 * ((2 * 1) * 3)$
34	$(3 / (1 / 4)) * 2$	124	$(2 * 4) * (1 * 3)$	124	$((1 * 2) * 3) * 4$	419	$((2 * 3) * 4) * 1$	614	$((4 * 2) * 3) * 1$	809	$(4 * (2 * 1)) * 3$
35	$3 / (1 / (4 * 2))$	125	$(2 * 4) * (1 * 3)$	125	$(1 * (2 * 3)) * 4$	420	$2 * ((3 * 4) * 1)$	615	$(4 * (2 * 3)) * 1$	810	$4 * ((2 * 1) * 3)$
36	$(4 / (1 / 3)) * 2$	126	$2 * ((4 / 1) * 3)$	126	$(1 * (2 * 3) * 4)$	421	$2 * ((3 * 4) * 1)$	616	$4 * ((2 * 3) * 1)$	811	$4 * ((2 * 1) * 3)$
37	$4 / (1 / (3 * 2))$	127	$(3 * (2 / 1)) * 4$	127	$1 * (2 * (3 * 4))$	422	$(2 * 3) * (4 * 1)$	617	$4 * (2 * (3 * 1))$	812	$4 * ((2 * 1) * 3)$
38	$(4 / (1 / 2)) * 3$	128	$(3 * (2 / 1)) * 4$	128	$(1 * 2) * (3 * 4)$	423	$(2 * 3) * (4 * 1)$	618	$(4 * 2) * (3 * 1)$	813	$(4 * (2 * 1)) * 3$
39	$4 / (1 / (2 * 3))$	129	$4 * ((2 / 1) * 4)$	129	$((1 * 2) * 4) * 3$	424	$((2 * 4) * 3) * 1$	619	$(4 * 2) * (3 * 1)$	814	$(4 * (2 * 1)) * 3$
40	$((2 / 1) * 3) * 4$	130	$(3 * (2 / 1)) * 4$	130	$(1 * 2) * (4 * 3)$	425	$(2 * 4) * (3 * 1)$	620	$(4 * 2) * (3 * 1)$	815	$(4 * (2 * 1)) * 3$
41	$(2 / 1) * (3 * 4)$	131	$(3 * (4 / 1)) * 2$	131	$(1 * (2 * 4) * 3)$	426	$2 * ((4 * 3) * 1)$	621	$4 * ((2 * 1) * 3)$	816	$4 * ((2 * 1) * 3)$
42	$((2 / 1) * 4) * 3$	132	$3 * ((4 / 1) * 2)$	132	$1 * ((2 * 4) * 3)$	427	$2 * ((4 * 3) * 1)$	622	$4 * ((2 * 1) * 3)$	817	$4 * ((2 * 1) * 3)$
43	$(2 / 1) * (4 * 3)$	133	$(4 * 2) * (1 * 3)$	133	$(1 * 2) * (4 * 3)$	428	$(2 * 4) * (3 * 1)$	623	$(4 * 2) * (1 * 3)$	818	$(4 * (2 * 1)) * 3$
44	$((3 / 1) * 2) * 4$	134	$(4 * (2 / 1)) * 3$	134	$((1 * 3) * 2) * 4$	429	$((2 * 4) * 1) * 3$	624	$(4 * 2) * (1 * 3)$	819	$(4 * (2 * 1)) * 3$
45	$(3 / 1) * (2 * 4)$	135	$4 * ((2 / 1) * 3)$	135	$(1 * (3 * 2)) * 4$	430	$(2 * 4) * (1 * 3)$	625	$(4 * 2) * (1 * 3)$	820	$(4 * (2 * 1)) * 3$

Gambar 4.5 Test Case User Input A 2 3 4





## **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](#)