LAPORAN TUGAS KECIL 1

IF-2211 - STRATEGI ALGORITMA

SEMESTER II 2022/2023

Zidane Firzatullah 13521163

A. Algoritma Brute Force

1. Melakukan loop untuk mencari permutasi angka dan operasi matematika. Loop ini mencari permutasi 4 angka dan 3 operasi matematika yang boleh sama dari 4 operasi matematika yang ada (penjumlahan (+), pengurangan (-), perkalian (x), dan pembagian (/).
2. Setelah didapatkan permutasi 4 angka dan 3 operasi matematika, misalnya w | x | y | z dengan huruf menyatakan angka dan | menyatakan operasi matematika, ekspresi tersebut akan diproses dengan 5 kemungkinan tanda kurung. Lima kemungkinan tanda kurung tersebut adalah: 1. ((w | x) | y) | z, 2. (w | (x | y)) | z, 3. (w | x) | (y | z), 4. w | ((x|y) | z), dan 5. w | (x (y | z)) .
3. Dengan 5 kemungkinan tanda kurung yang sebelumnya sudah dideskripsikan, akan dilakukan pemeriksaan apakah ekspresi tersebut sudah pernah dihitung sebelumnya dengan cara membuat sebuah struktur data hashset sebagai histori ekspresi yang sudah pernah dihitung dengan cara menyimpan ekspresi yang disimpan dalam bentuk string. Jika ekspresi sudah pernah dihitung, maka proses penghitungan akan dibatalkan.
4. Jika ekspresi yang diproses tidak ada dalam hashset, maka akan dilakukan penghitungan. Ekspresi bentukan dari permutasi angka, operasi matematika, dan tanda kurung tersebut akan dihitung dan apabila menghasilkan angka 24 ekspresi akan dianggap sebagai solusi yang valid dan string ekspresi akan di-*insert* ke dalam hashset.

B. Source Program

class: Main

```
import java.io.IOException;

public class Main {

   public static void main(String[] args) throws IOException {
       IoHandler.start();
   }

}
```

class: IoHandler

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Scanner;

import static java.lang.System.out;

public class IoHandler {

   private static HashMap<String, Integer> inputMap;

   public static void start() throws IOException {
       Scanner scanner = new Scanner(System.in);
       HashSet<String> exprs;

       while (true) {
           Integer[] inputs;
           out.println("generate random number? (Y/Any Key)");
           String randC = scanner.nextLine();
           boolean rand = randC.equalsIgnoreCase("y");

           try {
               inputs = getArray(rand);
           } catch (RuntimeException e) {
               out.println("invalid input.");
               continue;
           }

           if (rand)
               out.printf("generated numbers: %d %d %d %d\n",
inputs[0], inputs[1], inputs[2], inputs[3]);

           exprs = Processor.process(inputs);

           out.println("simpan ke file? (Y/Any Key) ");
           String input = scanner.next();

           if (input.equalsIgnoreCase("y")) {
               saveToFile(exprs);
           }

           break;
       }

   }

   private static Integer[] getArray(boolean random) {
       if (random) {
           return new Integer[]{MathUtil.getRandom(1, 14),
MathUtil.getRandom(1, 14),
                   MathUtil.getRandom(1, 14),
MathUtil.getRandom(1, 14)};
       }

       Scanner scanner = new Scanner(System.in);
       int i = 0;
       Integer[] arr = new Integer[4];
       out.println("input numbers (format: O O O O):");

       for (String s: scanner.nextLine().split(" ")) {
           int el = inputMap.getOrDefault(s, -1);

           if (el == -1 || i > 3)
               throw new RuntimeException("");

           arr[i] = el;
           ++i;
       }

       if (i != 4)
           throw new RuntimeException("");

       return arr;
   }

   private static void saveToFile(HashSet<String> exprs)
throws IOException {
       String url = "res.txt";

       BufferedWriter out = new BufferedWriter(new
FileWriter(url));
```

```
       out.write(String.format("%d solutions found.\n",
exprs.size()));
       for (String expr : exprs) {
           out.write(expr);
           out.newLine();
       }

       out.close();
   }

   static {
       inputMap = new HashMap<>();
       inputMap.put("A", 1);
       inputMap.put("J", 11);
       inputMap.put("Q", 12);
       inputMap.put("K", 13);
       inputMap.put("2", 2);
       inputMap.put("3", 3);
       inputMap.put("4", 4);
       inputMap.put("5", 5);
       inputMap.put("6", 6);
       inputMap.put("7", 7);
       inputMap.put("8", 8);
       inputMap.put("9", 9);
       inputMap.put("10", 10);
   }

}
```

class: MathUtil

```
import java.util.Random;

public class MathUtil {

   private static final Random random = new Random();

   public static Integer getRandom(int min, int max) {
       return random.nextInt(max - min) + min;
   }

}
```

class: Processor

```
import java.util.HashSet;

import static java.lang.System.nanoTime;
import static java.lang.System.out;
```

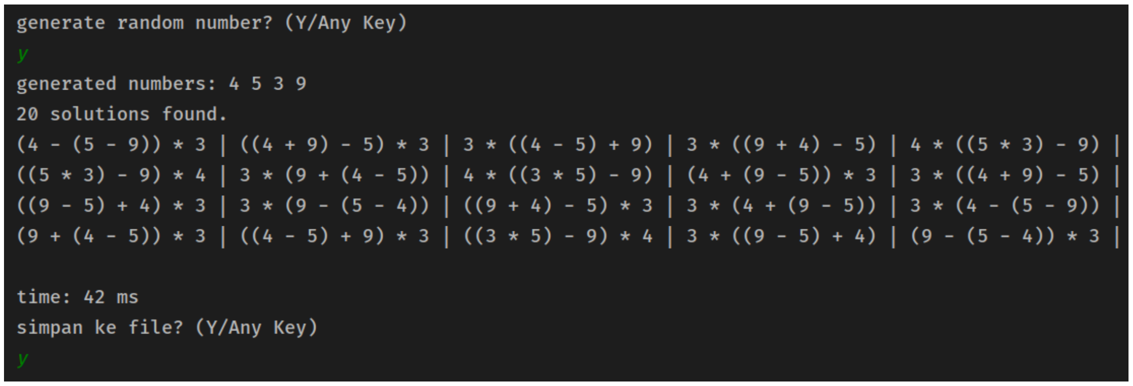
```
public class Processor {

   public static HashSet<String> process(Integer[] ln) {
       if (ln.length != 4)
           throw new RuntimeException("Call args error.");

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

       HashSet<Integer> lnSet;
       HashSet<String> history = new HashSet<>();

       int occ = 0;
       long startTime = nanoTime();

       for (int i = 0; i < 4; ++i) {
           lnSet = new HashSet<>();
           lnSet.add(i);

           for (int j = 0; j < 4; ++j) {
               for (int k = 0; k < 4; ++k) {
                   if (lnSet.contains(k))
                       continue;
                   lnSet.add(k);

                   for (int l = 0; l < 4; ++l) {
                       for (int m = 0; m < 4; ++m) {
                           if (lnSet.contains(m))
                               continue;
                           lnSet.add(m);

                           for (int n = 0; n < 4; ++n) {
                               for (int o = 0; o < 4; ++o) {
                                   if (lnSet.contains(o))
                                       continue;

                                   occ += countAll(ln[i],
ln[k], ln[m], ln[o], ops[j], ops[l], ops[n], history);

                               }
                           }
                           lnSet.remove(m);
                       }
                   }
                   lnSet.remove(k);
               }
           }
       }

       long finalTime = nanoTime() - startTime;
```

```
        out.printf("%d solutions found.\n", occ);

        int pIdx = 0;

        for (String key: history) {
            out.print(key + " | ");
            pIdx += 1;

            if (pIdx % 5 == 0)
                out.println();
        }

        out.printf("\ntime: %d ms\n", finalTime/1000000);

        return history;
    }

    private static int count(int num1, int num2, int num3, int
num4, char ops1, char ops2, char ops3,
                            HashSet<String> history) {
        String expr = String.format("((%d %c %d) %c %d) %c %d",
num1, ops1, num2, ops2, num3, ops3, num4);

        if (history.contains(expr))
            return 0;

        double value =
calculateExpression(calculateExpression(calculateExpression(nu
m1, num2, ops1), num3, ops2), num4, ops3);

        if (value >= 23.999 && value <= 24.001) {
            history.add(expr);
            return 1;
        }

        return 0;
    }

    private static int countSec(int num1, int num2, int num3,
int num4, char ops1, char ops2, char ops3,
                               HashSet<String> history) {
        String expr = String.format("(%d %c (%d %c %d)) %c %d",
num1, ops1, num2, ops2, num3, ops3, num4);

        if (history.contains(expr))
            return 0;

        double value =
calculateExpression(calculateExpression(num1,
calculateExpression(num2, num3, ops2), ops1), num4, ops3);
```

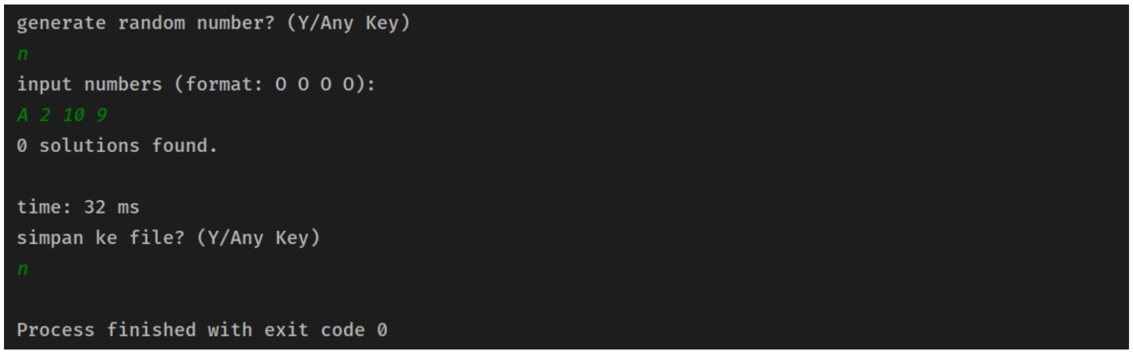
```
        if (value >= 23.999 && value <= 24.001) {
            history.add(expr);
            return 1;
        }

        return 0;
    }

    private static int countThi(int num1, int num2, int num3,
int num4, char ops1, char ops2, char ops3,
                               HashSet<String> history) {
        String expr = String.format("(%d %c %d) %c (%d %c %d)",
num1, ops1, num2, ops2, num3, ops3, num4);

        if (history.contains(expr))
            return 0;

        double value =
calculateExpression(calculateExpression(num1, num2, ops1),
calculateExpression(num3, num4, ops3), ops2);

        if (value >= 23.999 && value <= 24.001) {
            history.add(expr);
            return 1;
        }

        return 0;
    }

    private static int countF(int num1, int num2, int num3, int
num4, char ops1, char ops2, char ops3,
                             HashSet<String> history) {
        String expr = String.format("%d %c ((%d %c %d) %c %d)",
num1, ops1, num2, ops2, num3, ops3, num4);

        if (history.contains(expr))
            return 0;

        double value = calculateExpression(num1,
calculateExpression(calculateExpression(num2, num3, ops2),
num4, ops3), ops1);

        if (value >= 23.999 && value <= 24.001) {
            history.add(expr);
            return 1;
        }

        return 0;
    }

    private static int countFi(int num1, int num2, int num3,
int num4, char ops1, char ops2, char ops3,
                              HashSet<String> history) {
       String expr = String.format("%d %c (%d %c (%d %c %d))",
num1, ops1, num2, ops2, num3, ops3, num4);

       if (history.contains(expr))
           return 0;

       double value = calculateExpression(num1,
calculateExpression(num2, calculateExpression(num3, num4,
ops3), ops2), ops1);

       if (value >= 23.999 && value <= 24.001) {
           history.add(expr);
           return 1;
       }

       return 0;
   }

   private static int countAll(int num1, int num2, int num3,
int num4, char ops1, char ops2, char ops3,
                               HashSet<String> history) {
       return count(num1, num2, num3, num4, ops1, ops2, ops3,
history) +
               countSec(num1, num2, num3, num4, ops1, ops2,
ops3, history) +
               countThi(num1, num2, num3, num4, ops1, ops2,
ops3, history) +
               countF(num1, num2, num3, num4, ops1, ops2,
ops3, history) +
               countFi(num1, num2, num3, num4, ops1, ops2,
ops3, history);
   }

   private static double calculateExpression(double num1,
double num2, char ops) {
       if (ops == '*')
           return num1 * num2;

       if (ops == '+')
           return num1 + num2;

       if (ops == '-')
           return num1 - num2;

       if (ops == '/')
           return num1/num2;

       return 0;
   }
```

```
}
```

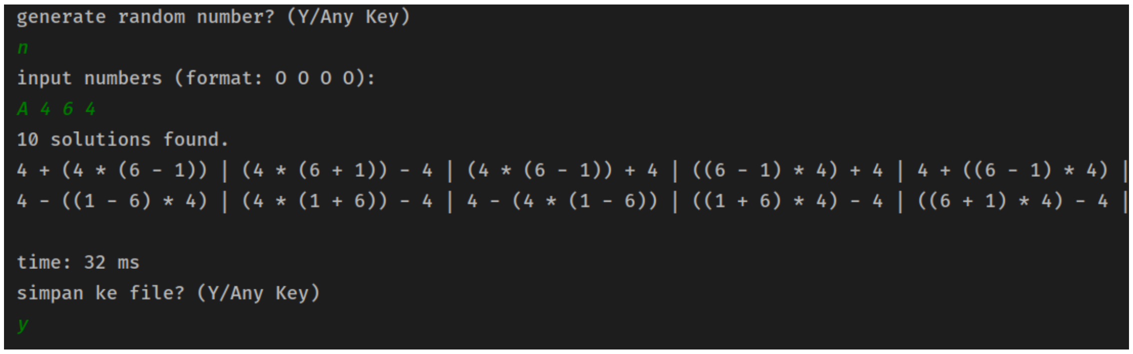
C. Screenshot Input dan Output



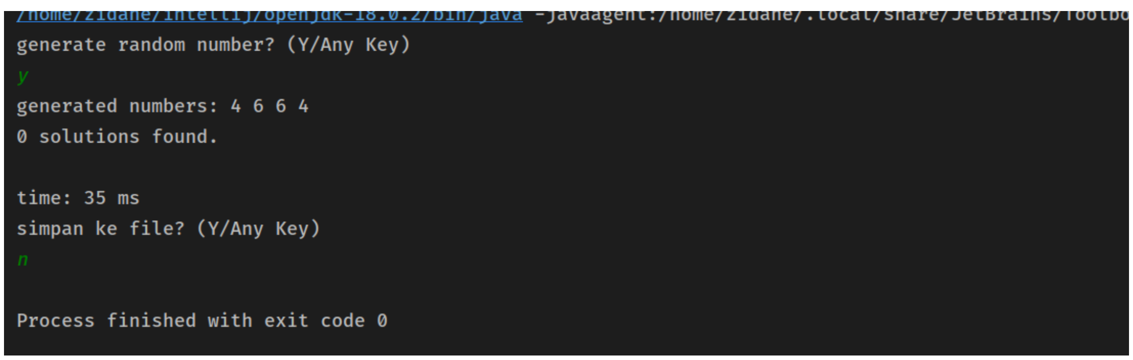
```
generate random number? (Y/Any Key)
y
generated numbers: 4 5 3 9
20 solutions found.
(4 - (5 - 9)) * 3 | ((4 + 9) - 5) * 3 | 3 * ((4 - 5) + 9) | 3 * ((9 + 4) - 5) | 4 * ((5 * 3) - 9) |
((5 * 3) - 9) * 4 | 3 * (9 + (4 - 5)) | 4 * ((3 * 5) - 9) | (4 + (9 - 5)) * 3 | 3 * ((4 + 9) - 5) |
((9 - 5) + 4) * 3 | 3 * (9 - (5 - 4)) | ((9 + 4) - 5) * 3 | 3 * (4 + (9 - 5)) | 3 * (4 - (5 - 9)) |
(9 + (4 - 5)) * 3 | ((4 - 5) + 9) * 3 | ((3 * 5) - 9) * 4 | 3 * ((9 - 5) + 4) | (9 - (5 - 4)) * 3 |

time: 42 ms
simpan ke file? (Y/Any Key)
y
```



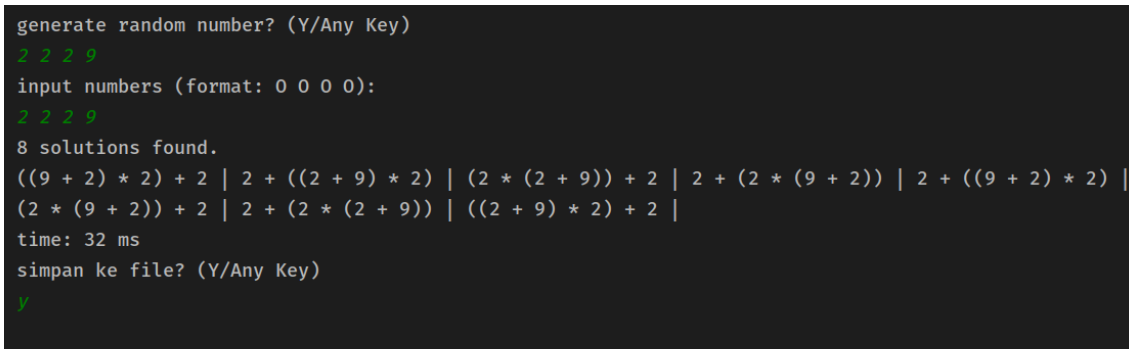
```
generate random number? (Y/Any Key)
n
input numbers (format: 0 0 0 0):
A 2 10 9
0 solutions found.

time: 32 ms
simpan ke file? (Y/Any Key)
n

Process finished with exit code 0
```



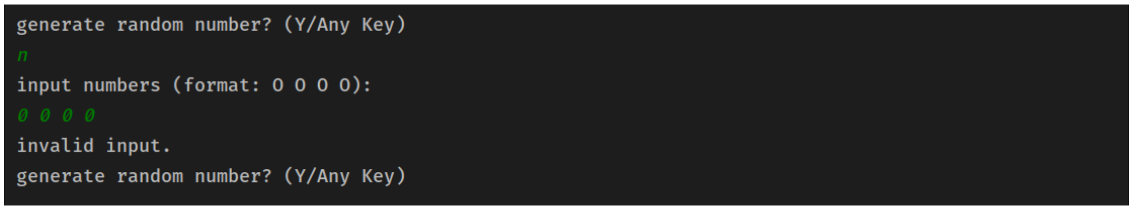
```
generate random number? (Y/Any Key)
n
input numbers (format: 0 0 0 0):
A 4 6 4
10 solutions found.
4 + (4 * (6 - 1)) | (4 * (6 + 1)) - 4 | (4 * (6 - 1)) + 4 | ((6 - 1) * 4) + 4 | 4 + ((6 - 1) * 4) |
4 - ((1 - 6) * 4) | (4 * (1 + 6)) - 4 | 4 - (4 * (1 - 6)) | ((1 + 6) * 4) - 4 | ((6 + 1) * 4) - 4 |

time: 32 ms
simpan ke file? (Y/Any Key)
y
```



```
/home/zidane/intellij/openjdk-18.0.2/bin/java -javaagent:/home/zidane/.local/share/JetBrains/Toolbo
generate random number? (Y/Any Key)
y
generated numbers: 4 6 6 4
0 solutions found.

time: 35 ms
simpan ke file? (Y/Any Key)
n

Process finished with exit code 0
```



```
generate random number? (Y/Any Key)
2 2 2 9
input numbers (format: 0 0 0 0):
2 2 2 9
8 solutions found.
((9 + 2) * 2) + 2 | 2 + ((2 + 9) * 2) | (2 * (2 + 9)) + 2 | 2 + (2 * (9 + 2)) | 2 + ((9 + 2) * 2) |
(2 * (9 + 2)) + 2 | 2 + (2 * (2 + 9)) | ((2 + 9) * 2) + 2 |
time: 32 ms
simpan ke file? (Y/Any Key)
y
```



```
generate random number? (Y/Any Key)
n
input numbers (format: 0 0 0 0):
0 0 0 0
invalid input.
generate random number? (Y/Any Key)
```

D. Link Repository Github

https://github.com/zidane-itb/Tucil1_13521163

E. Tabel

No	Deskripsi	Y/N
1	Program berhasil dikompilasi tanpa kesalahan	Y
2	Program berhasil running	Y
3	Program dapat membaca input / generate sendiri dan memberikan luaran	Y
4	Solusi yang diberikan program memenuhi	Y

	(berhasil mencapai 24)	
5	Program dapat menyimpan solusi dalam file teks	Y