ALGORITHM-DESIGN-TECHNIQUE: Bruteforce

STUDENT REGISTRATION ID (NRP): _____
NAME: _____
CLASS: _____

```
String[] components = {"+", "-", "*", "4", "5", "8", "3"};
```

ACTIVITY

Suppose you have an array of string containing "+", "-", "*", "4", "5", "8", "3"

*** How many ways to arrange the array into a single string?**
*** How many ways to arrange the array into a single string that represent a valid mathematics formula**
*** How many ways to arrange the array into a single string that represent a valid mathematics formula so that the result is "15"**
*** Write a piece of program/pseudo code to show all possible combination**

ALGORITHM-DESIGN-TECHNIQUE: Greedy

STUDENT REGISTRATION ID (NRP): _____
NAME: _____
CLASS: _____

ACTIVITY

1. Install this app in your phone: https://play.google.com/store/apps/details?id=com.saqgroup.algorithmpro&rdid=com.saqgroup.algorithmpro (You can use `Algorithm Pro` as keyword)

2. Read the `knapsack problem` section

3. Suppose you are participating in a reality show. The rule is quite simple. A knapsack is given to every participant, and each participant should put items as possible into the knapsack. Each item has their own weight and value. However, the knapsack capacity is just 0.5 Kg. Given the items as follow, which ones will you put into the knapsack?

Item	Weight	Value
A	100 g	Rp. 200.000,-
B	11 g	Rp. 25.000,-
C	50 g	Rp. 50.000,-
D	40 g	Rp. 30.000,-
E	100 g	Rp. 190.000,-
F	300 g	Rp. 200.000,-
G	200 g	Rp. 150.000,-
H	50 g	Rp. 20.000,-
I	80 g	Rp. 60.000,-
J	90 g	Rp. 40.000,-

4. Write a program/pseudocode to choose the best combination

ALGORITHM-DESIGN-TECHNIQUE: Divide & Conquer

STUDENT REGISTRATION ID (NRP): _____
NAME: _____
CLASS: _____

ACTIVITY

5 (next pivot)	3	2	4	7	1	10	6	8	9
3 (next pivot)	2	4	1	5 (pivot)	7 (next pivot)	10	6	8	9
2 (next pivot)	1	3 (pivot)	4 (next pivot)	5					
1 (next pivot)	2 (pivot)	3	4 (pivot)	5					
1 (pivot)	2	3	4	5					
1	2	3	4	5					

* Complete the table above
* How many steps are required to sort the numbers (compare it with bubble force)?
* How is the sorting algorithm related to `divide and conquer` principle?
* Write a program/pseudocode to do the sorting algorithm

ALGORITHM-DESIGN-TECHNIQUE: Decrease & Conquer

STUDENT REGISTRATION ID (NRP): _____
NAME: _____
CLASS: _____

ACTIVITY

10	1	8	7	4	5	6	9	3	2
1	8	7	4	5	6	**9**	3	2	**10**
1	8	7	4	5	6	3	2	**9**	**10**

1. Complete the table above
2. For the data above, which sorting algorithm is the fastest:
    * Bubble Sort needs _____ steps
    * Quick Sort needs _____ steps
    * Insertion Sort needs _____ steps
3. Find out best cases for:
    * Brute Force
    * Greedy
    * Divide and Conquer
    * Decrease and Conquer

ALGORITHM-DESIGN-TECHNIQUE: Back Tracking

STUDENT REGISTRATION ID (NRP): _____
NAME: _____
CLASS: _____

ACTIVITY

* Complete the table to find the solution for 4x4 sudoku
* Is it better than brute force? Why or why not?

<table>
<tr>
<td rowspan="10">1 2 _ _
_ _ 2 1
_ _ _ _
_ _ _ _
INITIAL CONDITION</td>
<td>1 2 1 _
_ _ 2 1
_ _ _ _
_ _ _ _
NOT OK</td>
<td></td>
</tr>
<tr>
<td>1 2 2 _
_ _ 2 1
_ _ _ _
_ _ _ _
NOT OK</td>
<td></td>
</tr>
<tr>
<td rowspan="8">1 2 3 _
_ _ 2 1
_ _ _ _
_ _ _ _
OK</td>
<td>1 2 3 1
_ _ 2 1
_ _ _ _
_ _ _ _
NOT OK</td>
</tr>
<tr>
<td>1 2 3 2
_ _ 2 1
_ _ _ _
_ _ _ _
NOT OK</td>
</tr>
<tr>
<td>1 2 3 3
_ _ 2 1
_ _ _ _
_ _ _ _
NOT OK</td>
</tr>
<tr>
<td>1 2 3 4
_ _ 2 1
_ _ _ _
_ _ _ _
OK</td>
</tr>
<tr>
<td rowspan="4">1 2 4 _
_ _ 2 1
_ _ _ _
_ _ _ _
OK</td>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</table>

ALGORITHM-DESIGN-TECHNIQUE: Back Tracking (BFS vs DFS)

STUDENT REGISTRATION ID (NRP): _____
NAME: _____
CLASS: _____

Suppose there is a tree with A as root and we are looking for H

<table>
<tr><td rowspan="6">A</td><td rowspan="2">B</td><td>E</td></tr>
<tr><td>F</td></tr>
<tr><td rowspan="2">C</td><td>G</td></tr>
<tr><td>H (*)</td></tr>
<tr><td rowspan="2">D</td><td>I</td></tr>
<tr><td>J</td></tr>
</table>

There are 2 approach to traverse the tree in order to find H.

- **DFS: Depth First Search (using stack)**
- **BFS: Breadth First Search (using queue)**

Here is an incomplete example of DFS approach:

Stack	Current Node	Traversed Nodes	Description
	A		Start from the root
B(A), C(A), D(A)	A		Get A's children, push them into stack
B(A), C(A), D(A)		A	Move A to traversed nodes
B(A), C(A)	D(A)	A	Pop D from the stack
B(A), C(A), I(DA), J(DA)	D(A)	A	Get D's children, push them into stack
B(A), C(A), I(DA), J(DA)		A, D(A)	Move D to traversed nodes
B(A), C(A), I(DA)	J(DA)	A, D(A)	Pop J From the stack

ACTIVITY

*** Find out how DFS works**
*** Complete the table**
*** Do BFS approach for the case above**
*** For solving sudoku, which one is better, DFS or BFS? Why?**
*** For solving rubik cube, which one is better, DFS or BFS? Why?**
*** What is your conclusion?**

ALGORITHM-DESIGN-TECHNIQUE: Dynamic Programming

STUDENT REGISTRATION ID (NRP): _____
NAME: _____
CLASS: _____

Try to understand the program below:

```
import java.util.HashMap;
public class DynamicProgrammingFibo {

  static long fibo (long number) {
    if (number < 3) {
      return 1;
    }
    return fibo(number - 1) + fibo(number - 2);
  }

  static HashMap<Long,Long> memo = new HashMap<Long,Long>();
  static long fiboDynamic (long number) {
    Long value = memo.get(number);
    if (value != null) {
      return value;
    }
    if (number < 3) {
      value = (long)1;
    } else {
      value = fiboDynamic(number - 1) + fiboDynamic(number - 2);
    }
    memo.put(number, value);
    return value;
  }

  public static void main (String[] args) {
    System.out.println("fiboDynamic(20) " + fiboDynamic(20));
    System.out.println("fibo(20) " + fibo(20));
    System.out.println("fiboDynamic(50) " + fiboDynamic(50));
    System.out.println("fibo(50) " + fibo(50));
  }

}
```

*** What is HashMap?**
*** What are the outputs of the program?**
*** Why fiboDynamic is faster than fibo?**