1. Implement the following scenario by **Bridge pattern** in Java :

Add the student names into the class register like “Ajay”, “Bala”, “Cathey”,”Chella”,”Dolly”, ”Ellan”, Francis”, ”Stella”.

Do the following operations on the class register

1. Display all student names
2. Delete “chella” from the register
3. Display previous and next names from the register.
4. Add “Zara” into the register
5. Display all student names after addition of “Zara”
6. Implement BubbleSort and Merge sort sorting algorithms using Strategy pattern

Input: 34,12,67,10,7

Output: 7,10,12,34,67

1. A Simple Scenario: As a developer I want to implement code so that it prints the numbers from 1 to 100.

Given: an input of numbers from 1–100  
When:  
A number is a multiple of ‘3’ return “Fizz”  
A number is a of ‘5’ return “Buzz”  
A number is a of both ‘3’ and ‘5’ return “FizzBuzz”  
A number is not divisible by ‘3’ or ‘5’ return the number itself  
Then: print “Fizz”, “Buzz”, “FizzBuzz” or the number accordingly

**Expected output:** 1, 2, Fizz, 4, Buzz, ……, 14, FizzBuzz, 16, …

Write the test cases and make all to be passed.

1. Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:
   1. Only one disk can be moved at a time.
   2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
   3. No disk may be placed on top of a smaller disk

**Sample Inpu**t:

* 3

**Sample Output:**

* Disk 1 moved from A to C
* Disk 2 moved from A to B
* Disk 1 moved from C to B
* Disk 3 moved from A to C
* Disk 1 moved from B to A
* Disk 2 moved from B to C
* Disk 1 moved from A to C

1. There are n gas stations along a circular route, where the amount of gas at the ith station is gas[i].

You have a car with an unlimited gas tank and it costs cost[i] of gas to travel from the ith station to its next (i + 1)th station. You begin the journey with an empty tank at one of the gas stations.

Given two integer arrays gas and cost, return *the starting gas station's index if you can travel around the circuit once in the clockwise direction, otherwise return* -1. If there exists a solution, it is **guaranteed** to be **unique**

**Example 1:**

**Input:** gas = [1,2,3,4,5], cost = [3,4,5,1,2]

**Output:** 3

**Explanation:**

Start at station 3 (index 3) and fill up with 4 unit of gas. Your tank = 0 + 4 = 4

Travel to station 4. Your tank = 4 - 1 + 5 = 8

Travel to station 0. Your tank = 8 - 2 + 1 = 7

Travel to station 1. Your tank = 7 - 3 + 2 = 6

Travel to station 2. Your tank = 6 - 4 + 3 = 5

Travel to station 3. The cost is 5. Your gas is just enough to travel back to station 3.

Therefore, return 3 as the starting index.

1. Given a string S of parentheses ‘(‘ or ‘)’. The task is to find a minimum number of parentheses ‘(‘ or ‘)’ (at any positions) we must add to make the resulting parentheses string is valid.

**Examples:** 

**Input:** str = "())"

**Output:** 1

One '(' is required at beginning.

Give your explanation for the above problem using Stack or Queue structure.

Find out the minimum number of parentheses needed to complete the input

1. ()[(){()}]
2. []{()()}
3. ((()))()()
4. ())((())
5. Write a program for AVL tree having functions for the following operations:

* Insert an element (no duplicates are allowed),
* Delete an existing element,
* Traverse the AVL (in-order, pre-order, and post-order)

**INPUT:**

Line 1 contains an integer NQ, the number of queries.

The next NQ lines contain queries and are of the form 'i x*x*' (Insert x*x* into an AVL tree) or 'd x*x*' (Delete x*x* from an AVL tree).

**OUTPUT:**

For each query, print value of an unbalanced node (if any) at which rotation is being applied.

The last three lines print 'Preorder traversal', 'Inorder traversal', and 'Postorder traversal' of an AVL tree that results after the execution of all NQ queries.

**SAMPLE INPUT:**

8

i 1

i 2

i 3

i 4

i 5

i 6

d 4

d 5

SAMPLE OUTPUT:

1

3

2

6

2 1 6 3

1 2 3 6

* + 1. 3 6 2

1. Write a Java program to find the sorted list by using Topological sorting [ Hint: apply DFS concept]

The first vertex in topological sorting is always a vertex with in-degree as 0 (a vertex with no incoming edges).



**Output:**

* + 1. 4 2 3 1 0 **or** 4 5 2 3 1 0

1. Write a Java program to find the K-th smallest element in BST.

Sample input: K=3

A picture containing clock

Description automatically generated

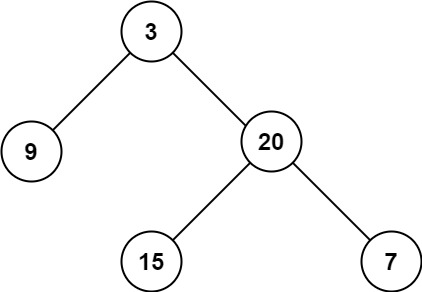
Sample Output:

10

1. Given the root of a binary tree and an integer targetSum, return true if the tree has a root-to-leaf path such that adding up all the values along the path equals targetSum.

A leaf is a node with no children.

**Example 1:**



**I**nput: root = [3,9,20,null,null,15,7]  
Output: 3