EXERCISE

1. Given N friends, each one can remain single or can be paired up with some other friend. Each friend can be paired only once. Find out the total number of ways in which friends can remain single or can be paired up.
Note: Since answer can be very large, return your answer mod 10^9+7.

Example:
Input: N = 3**Output:** 4**Explanation**:{1}, {2}, {3} : All single{1}, {2,3} : 2 and 3 paired but 1 is single.{1,2}, {3} : 1 and 2 are paired but 3 is single.{1,3}, {2} : 1 and 3 are paired but 2 is single.Note that {1,2} and {2,1} are considered same.

2. Given a gold mine of n*m dimensions. Each field in this mine contains a positive integer which is the amount of gold in tons. Initially the miner is at first column but can be at any row. He can move only (right->,right up /,right down\) that is from a given cell, the miner can move to the cell diagonally up towards the right or right or diagonally down towards the right. Find out maximum amount of gold he can collect.

Example:
Input : mat[][] = {{1, 3, 3}, {2, 1, 4}, {0, 6, 4}};Output : 12 {(1,0)->(2,1)->(1,2)}Input: mat[][] = { {1, 3, 1, 5}, {2, 2, 4, 1}, {5, 0, 2, 3}, {0, 6, 1, 2}};Output : 16(2,0) -> (1,1) -> (1,2) -> (0,3) OR(2,0) -> (3,1) -> (2,2) -> (2,3)Input : mat[][] = {{10, 33, 13, 15}, {22, 21, 04, 1}, {5, 0, 2, 3}, {0, 6, 14, 2}};Output : 83

3. Given an integer **K** and a queue of integers, we need to reverse the order of the first **K** elements of the queue, leaving the other elements in the same relative order.

Only following standard operations are allowed on queue.

- enqueue(x) : Add an item x to rear of queue
- dequeue() : Remove an item from front of queue
- size() : Returns number of elements in queue.
- front() : Finds front item.

Example:
Input:5 31 2 3 4 5**Output:** 3 2 1 4 5**Explanation:** After reversing the given input from the 3rd position the resultant output will be 3 2 1 4 5.

4. Given a string S. The task is to print all permutations of a given string.

Examples:

Input: ABSG**Output:**ABGS ABSG AGBS AGSB ASBG ASGB BAGS BASG BGAS BGSA BSAG BSGA GABS GASB GBAS GBSA GSAB GSBA SABG SAGB SBAG SBGA
SGAB SGBA**Explanation:**Given string ABSG has 24 permutations.

5. Given the root of a binary tree. Check whether it is a BST or not.
Note: We are considering that BSTs cannot contain duplicate Nodes.
A **BST** is defined as follows:
- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

Example:

Input: 2 / \1 3**Output:** 1 **Explanation:** The left subtree of root node contains node with key lesser than the root node’s key and the right subtree of root node contains node with key greater than the root node’s key. Hence, the tree is a BST.

6. Given a sorted array **arr** containing **n** elements with possibly duplicate elements, the task is to find indexes of first and last occurrences of an element **x** in the given array.

Example:

Input:n=9, x=5arr[] = { 1, 3, 5, 5, 5, 5, 67, 123, 125 }**Output:** 2 5**Explanation**: First occurrence of 5 is at index 2 and last occurrence of 5 is at index 5.