Assignment 4

Problem 1: PSG's Champions League Domino Strategy

PSG has advanced to the Champions League final and is tasked with setting up a 2×n domino-themed fan display at the stadium. The display consists of a rectangular grid of size 2×n. PSG fans can cover the grid using unlimited 2×1 dominoes, which represent their unity and strength in overcoming every obstacle on the road to the title. The challenge is to determine how many different ways PSG fans can completely cover the 2×n grid using these 2×1 dominoes.

Input Format:

A single integer `n`, representing the length of grid.

Constraints:

- $1 \leq n \leq 10000000$
- Time Limit: 1s
- Memory Limit: 4Mb

Output Format:

Return the number of distinct ways PSG fans can cover the grid using 2×1 dominoes, modulo 1e9+7(means %(1e9+7))

Example 1:

```
Input: 3
Output:3
```

Copy

Example 2:

```
Input: 75
Output:598991529
```

Copy

Explanation:

For `n = 3`, the possible ways to tile the grid are:

- Three vertical dominoes.
- One horizontal domino at the top and two vertical dominoes below.
- Two vertical dominoes at the top and one horizontal domino at the bottom.

Problem 2: The Battle of Hogwarts – Protect the Castle!

During the final battle at Hogwarts, Harry Potter, Hermione Granger, and Ron Weasley are entrusted with the mission to defend Hogwarts Castle from Voldemort's forces. The trio must strategically position their allies (represented as magical pieces: Rooks, Bishops, and Queens) on the battlefield (an N x N grid). However, there are magical barriers (watchmen wizards like the Order of the Phoenix) patrolling certain areas of the battlefield, and their paths must be avoided. Your task is to help Harry, Hermione, and Ron position their forces on the grid, ensuring that the total number of pieces placed on the board equals a given number n. Additionally, no piece can be placed where a watchman wizard stands, and no two pieces can attack each other.

Battle Ground Rules:

THERE CAN EXIST ONLY A SINGLE PIECE IN A ROW, YOU CANT PUT 2 BISHOPS or 2 QUEENS/ROOKS (seperated by a watchman) IN A ROW

- Board Size: You are given an N x N grid that represents the battlefield at Hogwarts.
- Magical Pieces:
  - Rook: Represents Harry's force, who can move horizontally or vertically.
  - Bishop: Represents Hermione's force, who can move diagonally.
  - Queen: Represents Ron's force, who can move horizontally, vertically, and diagonally.
- The type of magical piece to be placed in a cell is determined by the Manhattan distance from the Room of Requirement (a reference point):
  - If the Manhattan distance from the reference point is a multiple of 3*k (with k = 3), place a Rook.
  - If the Manhattan distance is 3*k + 1, place a Bishop.
  - Otherwise, place a Queen.
- Watchmen Wizards:
  - k watchmen wizards are patrolling the battlefield. They protect certain areas, so no magical piece can be placed where a watchman is located.

- Additionally, if a watchman wizard is located between two pieces on the same path (row, column, or diagonal), they prevent those pieces from attacking each other.

Goal:

Your goal is to place a combination of Rooks, Bishops, and Queens on the grid such that their total count equals number of rows(or number of columns) N, ensuring that:

- No two pieces can attack each other.
- Pieces avoid the watchmen's patrolling zones.

Input Format:

- An integer N representing the size of the battlefield (grid size N x N).
- An integer k representing the number of watchmen wizards.
- k pairs of integers (watchman_x, watchman_y) for the positions of the watchmen wizards.
- Two integers (ref_x, ref_y) for the Room of Requirement's position (used to calculate Manhattan distances)

Output Format:

The number of valid configurations where exactly N pieces (Rooks, Bishops, Queens) can be placed safely on the board, avoiding attacks and watchmen zones

Constraints:

- $1 \leq N \leq 12$
- $0 \leq k \leq N*N$
- $0 \leq watchman\_x < N$
- $0 \leq watchman\_y < N$
- $0 \leq ref\_x \leq N$
- $0 \leq ref\_y \leq N$
- Time Limit: 1s
- Memory Limit: 2Mb

Examples:

```
Input:
```

```
3
1
0 2
0 0
```

```
Output:
1
```

Explanation:

The only possible configuration is
Rook Empty Watchman
Empty Empty Rook
Empty Rook Empty

```
Input:
4
0
0 0
```

```
Output:
4
```

Problem 3: Sorting by Repeated Removal of Maximums

You are given an array A of length N. Your task is to sort the array, but you can only perform the following operation:

Remove the largest element from the array and append it to the end of the non-frozen elements. Once an element is moved, it becomes "frozen"—meaning it cannot be considered again or shifted. Repeat this process, ensuring that at each step, only the non-frozen elements are considered, and the removed element is appended to the end of the non-frozen portion of the array.

The goal is to construct a sorted array using minimum number of these repeated “remove largest” operations. If it is not possible to construct a sorted array with these operations, print -1.

Rules:

- You can remove the largest element from the current array only if it's uniquely the largest. If there are multiple largest elements, you cannot perform the removal and then print -1.

Input Format:

- The first line contains a single integer N, the length of the array.
- The second line contains N space-separated integers A[1], A[2], ..., A[N] representing the elements of the array.

Output Format:

Print a single integer -1 if it's not possible to construct a sorted array using the above operation. Otherwise, print the sorted array in ascending order.

Constraints:

- 1 ≤ N ≤ 1000
- -10^6 ≤ A[i] ≤ 10^6
- Time Limit: 1s
- Memory Limit: 4Mb

Do not use any sorting techniques (bubble sort,insertion sort etc.) except the one given in the question. Deltetion and appending should happen in the same array.

Examples:

```
Input:
5
3 2 5 1 4

Output:
1 2 3 4 5
```

```
Input:
10
-664657 86814 -867638 -996301 983931 -890367 -373178 129237 111187 -253255

Output:
-996301 -890367 -867638 -664657 -373178 -253255 86814 111187 129237 983931
```