Q1. Continuous Sum Query

There are **A** beggars sitting in a row outside a temple. Each beggar initially has an empty pot. When the devotees come to the temple, they donate some amount of coins to these beggars. Each devotee gives a fixed amount of coin(according to their faith and ability) to some **K** beggars sitting next to each other.

Given the amount **P** donated by each devotee to the beggars ranging from **L** to **R** index, where **1** <= **L** <= **R** <= **A**, find out the final amount of money in each beggar's pot at the end of the day, provided they don't fill their pots by any other means.
For ith devotee B[i][0] = L, B[i][1] = R, B[i][2] = P, Given by the 2D array **B**

Input 1:-

```
A = 5
B = [[1, 2, 10], [2, 3, 20], [2, 5, 25]]
```

Output 1:-

```
10 55 45 25 25
```

Explanation 1:-

First devotee donated 10 coins to beggars ranging from 1 to 2. Final amount in each beggars pot after first devotee: [10, 10, 0, 0, 0]
Second devotee donated 20 coins to beggars ranging from 2 to 3. Final amount in each beggars pot after second devotee: [10, 30, 20, 0, 0]
Third devotee donated 25 coins to beggars ranging from 2 to 5. Final amount in each beggars pot after third devotee: [10, 55, 45, 25, 25]

Q2. Prefix maximum

Kamal is a software developer and he's working on a new feature for an e-commerce website. The website has a list of prices for different products, and Kamal needs to find the maximum price of all products up to a given index.

He has the list of prices in an array **A** of length **N**, and he needs to write a program that will return the maximum price occurring in the subarray from 0 to i for every index i. Kamal needs your help to implement this function.

Input 1:

```
A = [9, 7, 6, 18, 2]
```

Input 2:

```
A = [16, 8, 24, 9, 25, 17]
```

Output 1:

```
[9, 9, 9, 18, 18]
```

Output 2:

```
[16, 16, 24, 24, 25, 25]
```

Q3. Suffix maximum

Given an array **A** of length **N**, For every index **i**, you need to find the maximum value in subarray from **i** to **N-1**.

Input 1:

```
A = [12, 5, 6, 7]
```

Input 2:

```
A = [15, 9, 7, 11, 10]
```

Output 1:

```
[12, 7, 7, 7]
```

Output 2:

```
[15, 11, 11, 11, 10]
```

Q4. Rain Water Trapped

Given a vector **A** of non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

Input 1:

```
A = [0, 1, 0, 2]
```

Input 2:

```
A = [1, 2]
```

Output 1:

```
1
```

Output 2:

```
0
```

Q5. Max Sum Contiguous Subarray

Find the maximum sum of **contiguous non-empty subarray** within an array **A** of length **N**.

Input 1:

```
A = [1, 2, 3, 4, -10]
```

Input 2:

```
A = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
```

Output 1:

```
10
```

Output 2:

```
6
```

Q6. Range Sum - II

You are given an integer array **A** of size **N**. You need to perform **Q** queries on the given array and return the **final array** after processing **all the queries**.

Each query is of the form (l, r, c), where **l** and **r** are **indices** (1-based) representing a range in the array **A**, and **c** is an integer value.

For each query, you are required to **add** the value **c** to every element within the range [l, r] (inclusive).

Input 1:

```
A = [1, 2, 1, 4]
B = [
     [2, 3, 2]
     [1, 4, 5]
     [4, 4, 1]
   ]
```

Input 2:

```
A = [1, 1, 2]
B = [
     [2, 3, 2]
     [1, 3, 5]
   ]
```

Output 1:

```
[6, 9, 8, 10]
```

Output 2:

```
[6, 8, 9]
```

Explanation 1:

```
First query, l = 2, r = 3, c = 2, after processing array becomes : [1, 4, 3, 4]
Second query, l = 1, r = 4, c = 5, after processing array becomes : [6, 9, 8, 9]
Third query, l = 4, r = 4, c = 1, after processing array becomes : [6, 9, 8, 10]
Final array is [6, 9, 8, 10]
```

Explanation 2:

```
First query, l = 2, r = 3, c = 2, after processing array becomes : [1, 3, 4]
Second query, l = 1, r = 3, c = 5, after processing array becomes : [6, 8, 9]
Final array is [6, 8, 9]
```

Q7. Chocolate Distribution

Given an array **A** of **N** integers where the i-th element represent the number of chocolates in the i-th packet.

There are **B** number of students, the task is to distribute chocolate packets following below conditions:

```
1. Each student gets one packets.
2. The difference between the number of chocolates given to any two students is minimum.
```

Return the minimum difference (that can be achieved) between the student who gets minimum number of chocolates and the student who gets maximum number of chocolates.

Note: If you can't give each student 1 packet, return 0.

Input:

```
A : [3, 4, 1, 9, 56, 7, 9, 12]
B : 5
```

Output:

```
6
```