4304-Data Structures Lab. Winter 2022
Date: November 15, 2022.
Target Group: All
Topic: Segment Tree

Instructions:

- Task naming format: fullID_T01L13_2A.c/CPP
- If you find any issues in the problem description/test cases, comment in the google classroom.
- If you find any tricky test case that I didn’t include but others might forget to handle, please comment! I’ll be happy to add.
- Use appropriate comments in your code. This will help you to recall the solution in the future easily.
- Obtained marks will vary based on the efficiency of the solution.
- Do not use <bits/stdc++.h> library, add the necessary libraries one after another.
- Do not use any built-in functions or any STL libraries. If the solution is extremely difficult without the usage of such components, discuss it with the instructors.
- Modified sections will be marked with BLUE color.

Group	**Tasks**
1B	1 2 3
2B	1 2 3
2A	1 2 3
1A	

Task-1:

Given an array with **N** elements (indexed from 1 to N) and a Query within range (i,j), your task is to find the minimum value of that range.

Input:
Each test case will have two values (N, Q) in the first line, denoting the total number of elements N and the total number of queries Q.
The following line will take N numbers (A segment tree will be built using these values).
Each of the following Q lines will have two values indicating i & j for each range.

Output:
For each Query, you have to print the minimum value of range (i,j).

Sample Input	**Sample Output**

5 3
78 1 22 12 3
1 2
3 5
4 4
	
	1
	3
	12

1 1
10
1 1
	
	10

6 6
20 50 10 40 90 30
1 6
3 3
5 5
5 6
4 6
3 6
	
	10
	10
	90
	30
	30
	10

Note: Maximum complexity of each query should be in O(logN).

Task 2:

Robin Hood likes to loot rich people and help poor people with this money. Instead of mixing the money, he keeps n sacks to keep money from different sources separately. The sacks are numbered 1 to N.
With these sacks, he usually performs three types of tasks:

1. Give all the money of i^{th} sack to the poor, leaving it empty.
2. Add a new component (given in input) into j^{th} sack.
3. Find the total amount of money from i^{th} sack to j^{th} sack.

Since he is not a programmer, he seeks your help.

Input:
Each test case will have two values N, Q in the first line denoting the total number of elements (N) and the total number of queries(Q).
The following line will take N numbers.
Each of the following Q lines can have values as:

- 1 i (Give all money from i^{th} sack to the poor and show current status.)
- 2 i v (Add v amount of money to i^{th} sack.)
- 3 i j (Find the total amount of money from i^{th} to j^{th} sack.)

Output:

- Type-1: Print the amount of money that the poor will receive.
- Type-2: Show the current status of the sacks after the update.
- Type-3: Print the total amount from i to j.

Sample Input	**Sample Output**
5 6	
3 2 1 4 5	
1 5	5 (3 2 1 4 0)
2 4 4	3 2 1 8 0
3 1 4	14
1 3	1 (3 2 0 8 0)
3 3 5	8
1 2	2 (3 0 0 8 0)

Link: https://lightoj.com/problem/curious-robin-hood

Task 3:

Implement the Query and Update operation for a **Min-Segment tree** maintaining the ‘Lazy-propagation’ property. Test the system for different cases and make sure everything is implemented correctly.

The first line contains N and Q denoting the number of elements and number of queries/updates. The following Q lines will contain query/update operations.

- Update is denoted by 2. Format: 2 i j x. You have to add x with the value that is present at the node.
- Query is denoted by 1. Format: 1 i j. You have to return the minimum value within the range (i, j)

Print the status of the segment tree and lazy tree after each operation.

Sample Input	**Sample Output**
8 4 -1 2 4 1 7 1 3 2	-1 -1 1 -1 1 1 2 -1 2 4 1 7 1 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 1 4 3	**1** **2** 1 -1 1 1 2 -1 2 4 1 7 1 3 2 0 0 0 **3** **3** 0 0 0 0 0 0 0 0 0 0
2 1 4 1	1 **3** 1 -1 1 1 2 -1 2 4 1 7 1 3 2 0 0 0 **4** **4** 0 0 0 0 0 0 0 0 0 0
2 2 2 2	1 **3** 1 **3** **5** 1 2 **3** **8** 4 1 7 1 3 2 0 0 0 **0** **0** 0 0 0 0 **4** **4** 0 0 0 0
1 4 6	1 1 3 1 3 5 1 2 3 8 **8** **5** 7 1 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0