PROBLEM STATEMENT

The optimal solution is to remove the last two elements to reduce x to zero

INPUT: Nums [] = [1, 1, 4, 2, 3]

INPUT: X = 5

a. Remove the leftmost element from the array nums

OR

b. Remove the rightmost element from the array nums

c. Subtract the its value from x.

Return the minimum number of operations to reduce x to exactly 0 if it's possible otherwise return -1.

2. ALGORITHM

ALGORITHM	**MINIMIUM OPERATIONS TO REDUCE X TO ZERO**
INPUT:	Integer array nums & Integer x. `(int[] nums, int x)`
OUTPUT:	Result of operation
STEP 1:	**START**
STEP 2:	Subtract the total number of nums from the value of x And save it in a variable – target `int target = Arrays.stream(nums).sum() - x;`
STEP 3:	Initialize the `numOperation` = -1 Initialize the current = 0 `int n = nums.length, numOperation = -1, current = 0;`
STEP 4:	Iterate through the array of nums starting at rightmost position `for (int right = 0, left = 0; right < n; right++)`

STEP 5:	Update the current position as it iterates `current += nums[right];`

STEP 6:	Continue the iteration:
	`while (current > target && left <= right)`
	And set the:
	`current -= nums[left++];`

STEP 7:	As it iterates Check if below condition is meet and get the `numOperation`:
	`if (current == target)`
	`numOperation = Math.max(numOperation, right - left + 1);`

STEP 8:	Return numOperation by subtracting nums.leght from wSize Else return -1
STEP 9:	**STOP**

3. CODE

```
package com.shedrack.assesment.solution;

import java.util.Arrays;

public class TestSolution {


public static void main(String[] args) {

            int[] nums = { 3, 2, 20, 1, 1, 3 };
            minOperations(nums, 10);
      }

public static int minOperations(int[] nums, int x) {
            int target = Arrays.stream(nums).sum() - x;
            int n = nums.length, numOperation = -1, current = 0;

            for (int right = 0, left = 0; right < n; right++) {
                  current += nums[right];
                  while (current > target && left <= right)
                        current -= nums[left++];

                  if (current == target)
                        numOperation = Math.max(numOperation, right - left + 1);
            }
            System.out.println(numOperation != -1 ? n - numOperation : -1);
            return numOperation != -1 ? n - numOperation : -1;
      }

}
```