ALGOS TASK 2

EXPLANATION OF CODES

1)MAX POSSIBLE SUM

- To find the max possible sum of elements, each element of A must be max.
- The first element of B is max of first 2 elements of A. Thus, the first element can be B[0]. The 2nd element can be B[0] only if B[1]>A[1].
- So, from the 2nd element onwards till the (n-1)th element, A[i] will be equal to the min of B[i-1] and B[i] asper the condition that B[i] should be greater than A[i] and A[i+1]
- For the last element, A[n] should be equal to B[n-1]
- The next for loop has the code to sum up all the elements in the array A.

2)FIND THE LENGTH

- Storing the number of proper bracket count for each case in an array-results.
- x-for counting the number of open brackets,<
- y-for counting number of closing brackets,>
- We don't want the cases when number of > are greater than number of <, so that condition is put in the while loop.
- j- for traversing through the input strings
- y-number of closing brackets which when equal to x gives the number of proper pairs of brackets.
- In cases like "<>>>"- y becomes more than x as it goes through the while loop 1 extra time, therefore, we need to subtract 1.
- In cases like "<<<>"- x will not be equal to y but still goes through the loop because x>=y is satisfied...hence in those cases, there is not even 1 proper bracket, so our answer will be 0.
- As per the challenge, we need to print total number of proper brackets, which will be 2*y

3)STAIRS PROBLEM

- Takes input for total number of stairs and broken stairs separated by space.
- broken is a set which will have integer user input.
- paths is an array where paths[i] stores the number of ways to reach the ith step.
- number of ways to reach the 0th step is defaulted to 1.
- Now, we observe that the number of ways to reach the ith step is the sum of the number of ways to reach the (i-1)th and (i-2)th steps. Because, there are 2 possibilities to reach the ith step, from i-1 or from (i-2)th step.
- If 'i' is an element the set, broken, i.e., if 'i' is a broken step, y=0 -> number of ways to reach that step=0. Then that gets stored in the array in the paths[i] position.
- If i is not a broken step, and if it is the first step, then number of ways to reach it are 1, otherwise they are the sum of the number of ways to reach the previous 2 steps.
- Since this number of ways can be very large, we take the modulo M=100000007 using [(a-b)%M]= (a%M + b%M)%M
- This sum gets appended to the array.
- Finally, it prints paths[n] which is the number of ways to reach the nth step.

4)MAX ELEMENT IN ARRAY

- Array a with elements a[k]=k is created. Since k must start from 1, x=k+1 is appended.
- q takes the input of number of queries/operations to be performed
- the left index, right index and value to be added to all elements in that range is then taken from user using split() which allocates space separated input values to the corresponding variables on the left.

- Since l, r are index values according to an array where 1st index is 1, Therefore, we start with l1=l-1 and end with r1=r-1 as our array index numbers start from 0.
- The next for loop traverses through the array for elements in the given range. The corresponding value is added to those elements.
- The above step takes place for all the q queries.
- Now, in our final array, to find the maximum element, we initially suppose the first element is max, then compare the max element with other elements, if we find a bigger element, that get substituted as the max element and the further elements are compared with it
- Finally prints the max element in the array

5)MAX ELEMENT IN ARRAY 2

- Array a with elements a[k]=k is created. Since k must start from 1, x=k+1 is appended.
- q takes the input of number of queries/operations to be performed
- the left index, right index and value to be added to all elements in that range is then taken from user using split() which allocates space separated input values to the corresponding variables on the left.
- Since l, r are index values according to an array where 1st index is 1, Therefore, we start with l1=l-1 and end with r1=r-1 as our array index numbers start from 0.
- The next for loop traverses through the array for elements in the given range. The corresponding value is added to those elements.
- The above step takes place for all the q queries.
- Now, in our final array, to find the maximum element, we initially suppose the first element is max, then compare the max element with other elements, if we find a bigger element, that get substituted as the max element and the further elements are compared with it
- Finally prints the max element in the array