Assignment 8

You may recall that an array arr is a **mountain array** if and only if:

- arr.length >= 3
- There exists some i with 0 < i < arr.length - 1 such that:
  - arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
  - arr[i] > arr[i + 1] > ... > arr[arr.length - 1]

Given a mountain array mountainArr, return the **minimum** index such that mountainArr.get(index) == target. If such an index does not exist, return -1.

You cannot access the mountain array directly. You may only access the array using a MountainArray interface:

- MountainArray.get(k) returns the element of the array at index k (0-indexed).
- MountainArray.length() returns the length of the array.

Submissions making more than 100 calls to MountainArray.get will be judged *Wrong Answer*. Also, any solutions that attempt to circumvent the judge will result in disqualification.

Example 1:

Input: mountainArr = [1,2,3,4,5,3,1], target = 3
Output: 2
Explanation: 3 exists in the array, at index=2 and index=5. Return the minimum index, which is 2.

Example 2:

Input: mountainArr = [0,1,2,4,2,1], target = 3
Output: -1
Explanation: 3 does not exist in the array, so we return -1.

Constraints:

- 3 <= mountainArr.length() <= 10_4
- 0 <= target <= 10_9
- 0 <= mountainArr.get(index) <= 10_9

Program:

```
class Solution {
    public int findInMountainArray(int target, MountainArray
mountainArr) {
        int n = mountainArr.length();
        int start=0;
        int end=n-1;
        while(start<end){
        int mid=start+(end-start)/2;
        int mid-val=mountainArr.get(mid);
        int nextval=mountainArr.get(mid+1);
        if (mid-val<nextval) {
        start=mid+1;
        }
        else{
           end=mid;
        }
        }
        Int peak=start;
        Int result=binarysearch(mountainArr,0,peak,target,true);
        If(result!=-1){
        return result;
        }
        return binarysearch(mountainArr,peak+1,n-1,target,false);
```

```
}
Public int binarysearch(mountainArray,int start,int end,int
target,boolean ascending){
While(start<=end){
Int mid=start +(end-start)/2;
Int midval=arr.get(mid);
If(midval==target){
return mid;

}
If(ascending){
If(target<midval){
End=mid-1;
}
else{
Start=mid+1;
}
}
else{
if(target>midval){
end=mid-1;
}
else{
start=mid+1;
}
}
}
return -1;
}
```