山脉数组中查找目标值

给你一个 **山脉数组** mountainArr，请你返回能够使得 mountainArr.get(index) **等于** target **最小** 的下标 index 值。

如果不存在这样的下标 index，就请返回 -1。

何为山脉数组？如果数组 A 是一个山脉数组的话，那它满足如下条件：

首先，A.length >= 3

其次，在 0 < i < A.length - 1 条件下，存在 i 使得：

- A[0] < A[1] < ... A[i-1] < A[i]
- A[i] > A[i+1] > ... > A[A.length - 1]

你将 **不能直接访问该山脉数组**，必须通过 MountainArray 接口来获取数据：

- MountainArray.get(k) - 会返回数组中索引为 k 的元素（下标从 0 开始）
- MountainArray.length() - 会返回该数组的长度

```
/**
 * // This is the MountainArray's API interface.
 * // You should not implement it, or speculate about its implementation
 * class MountainArray {
 *   public:
 *     int get(int index);
 *     int length();
 * };
 */

class Solution {
public:
    int findInMountainArray(int target, MountainArray &mountainArr) {
        //先找到峰值
        int left=0;
        int right=mountainArr.length()-1;
        while(left<right)
        {
            int mid=left+(right-left)/2;
            if(mountainArr.get(mid)<mountainArr.get(mid+1))
            {
                left=mid+1;//[mid+1,right]
```

```
            }
            else
            {
                right=mid;//[left,mid]
            }
        }
        int top=left;

        //峰值左侧二分查找
        int l=0;
        int r=top+1;
        while(l<r)
        {
            int mid=l+(r-l)/2;
            if(mountainArr.get(mid)==target)
                return mid;
            else if(mountainArr.get(mid)<target)
                l=mid+1;
            else
                r=mid;
        }

        //峰值右侧二分查找
        int x=top-1;
        int y=mountainArr.length();
        while(x<y)
        {
            int mid=x+(y-x)/2;
            if(mountainArr.get(mid)==target)
                return mid;
            else if(mountainArr.get(mid)>target)
                x=mid+1;
            else
                y=mid;
        }
        return -1;
    }
};
```