DAY 21

INTERVIEW BIT PROBLEMS :

1. Nearest Smaller Element

Given an array, find the nearest smaller element G[i] for every element A[i] in the array such that the element has an index smaller than i.
More formally,

G[i] for an element A[i] = an element A[j] such that
j is maximum possible AND
j < i AND
A[j] < A[i]

Elements for which no smaller element exist, consider next smaller element as -1.

Input Format
The only argument given is integer array A.
Output Format
Return the integar array G such that G[i] contains nearest smaller number than A[i].If no such element occurs G[i] should be -1.

For Example
Input 1:
A = [4, 5, 2, 10, 8]
Output 1:
G = [-1, 4, -1, 2, 2]
Explaination 1:
index 1: No element less than 4 in left of 4, G[1] = -1
index 2: A[1] is only element less than A[2], G[2] = A[1]
index 3: No element less than 2 in left of 2, G[3] = -1
index 4: A[3] is nearest element which is less than A[4], G[4] = A[3]
index 4: A[3] is nearest element which is less than A[5], G[5] = A[3]

Input 2:
A = [3, 2, 1]
Output 2:
[-1, -1, -1]
Explaination 2:
index 1: No element less than 3 in left of 3, G[1] = -1
index 2: No element less than 2 in left of 2, G[2] = -1
index 3: No element less than 1 in left of 1, G[3] = -1

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @return a list of integers
```

```
    def prevSmaller(self, A):
        stack=[-1]
        out=[]
        n=len(A)
        for i in range(n):
            while n>0 and stack[-1]>=A[i]:
                stack.pop()
            if n==0:
                out.append(-1)
            else:
                out.append(stack[-1])
            stack.append(A[i])
        return out
```

C++

```
vector<int> Solution::prevSmaller(vector<int> &A) {
    //create an empty stack
    stack<int>Stack;
    Stack.push(-1);
    int n;
    n=A.size();
    vector<int>output;
    //Traversing all array elements
    for(int i=0;i<n;i++){
        while(n>0 && Stack.top()>=A[i]){
            Stack.pop();
        }
        if(n==0){
            output.push_back(-1);
        }
        else{
            output.push_back(Stack.top());
        }
        Stack.push(A[i]);
    }
    return output;
}
```