Todays Content:

Attendance = 22

1. Nearest smaller element in left
2. Nearest smaller element in left

1st smaller element in left side:

Given ar[N], for every ar[i] : find nearest smaller element in leftside
Nearest ele < ar[i], distance between indices

Eg1

	0	1	2	3	4	5
ar[6] =	4	5	2	10	3	12
ans[6] =	-1	4	-1	2	2	3

Eg2

	0	1	2	3	4	5	6	7
ar[8] =	4	6	10	11	7	8	3	5
ans[8] =	-1	4	6	10	6	7	-1	3

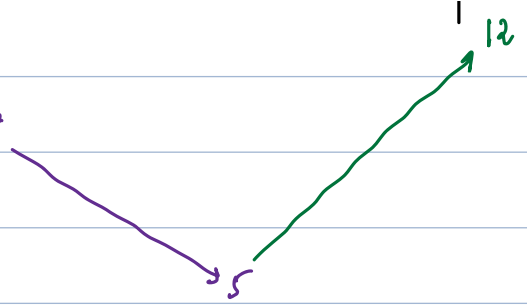
Ideal: for every ar[i]:
Iterate in left: find 1st element < ar[i].
0 .. i-2 i-1 i

```
vector<int> nearestElement(int ar[], int N){ TC: O(N^2) SC: O(1)
    vector<int> ans(N,-1);

    for(int i=0; i<N; i++){
        #for every ar[i] calculate 1st ele < ar[i] in left
        for(int j=i-1; j>=0; j--){        3 2 14 6 13 12 10
            if(ar[j] < ar[i]){
                ans[i] = ar[j];
                break;
            }
        }
    }
    return ans;
}
```

Idea2

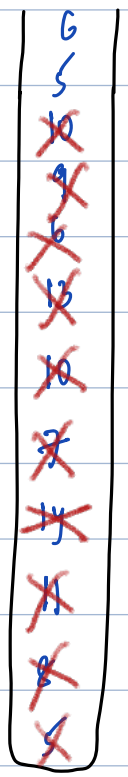
#Hint: { 8 14 11 10 5 11 } 8



With presence of 5, 8 cannot be ans for future elements.

#obs:

	0	1	2	3	4	5	6	7	8	9	10	11
ar[12] :	5	8	11	14	7	10	13	6	9	10	5	6
ans[12] :	-1	5	8	11	5	7	10	5	6	9	-1	5



#Containers:

1. Any element which can be ans, store it
2. If we are 100% sure, it cannot be ans, delete it.

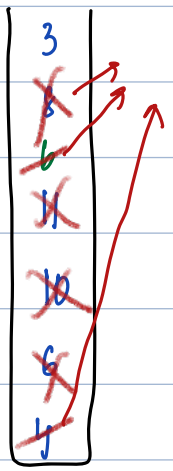
Operations:

1. Insert top()
2. delete top()
3. Ans top()
4. Size()

#Ans: Stack datastructure

Dry Run2:

	0	1	2	3	4	5	6	7
ar[8] =	4	6	10	11	6	8	3	5
ans[] =	-1	4	6	10	4	6	-1	3




```
stack<int> st;
  element ar[i] comes:
while( st.size() > 0 && st.top() >= ar[i]){
}   st.pop();
if (st.size() > 0){
}  ans[i] = st.top();
else{
}  ans[i] = -1
st.push(ar[i])
```

```
vector<int>  nearestElement(vector<int> &ar) { TC: O(N)  SC: O(N)
    stack<int> st;
    int N = ar.size();
    vector<int> ans(N, -1);
    for(int i=0; i<N; i++) {
        while( st.size() > 0 && st.top() >= ar[i]){
        }   st.pop();
        if (st.size() > 0){
        }  ans[i] = st.top();
        else{
        }  ans[i] = -1
        st.push(ar[i])
    }
    return ans;
}
```

Total Iterations = N + N = 2N

Total Outer loop + Total Inner loop.

(N)

1 pop = 1 inner iteration.

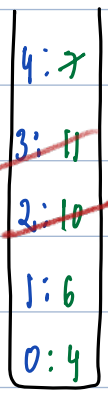
Total push = N

Total pop = N

Total Inner loop = N

Nearest smaller Index in left:

	0	1	2	3	4	5	6	7
ar[8] =	4	6	10	11	7	8	3	5
ans[8] =	-1	0	1	2	1			

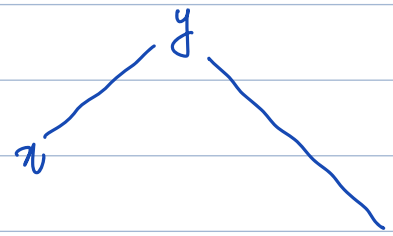


7 # Containers:

1. Any element which can be ans, store its index.
2. If we are 100% sure, it cannot be ans, delete it.

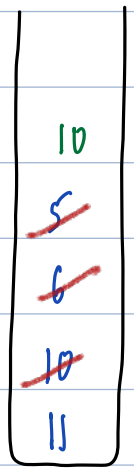
```
vector<int> nearestElementIndex(vector<int> ar) {
    stack<int> st;
    int n = ar.size();
    vector<int> ans(n, -1);
    for(int i=0; i<n; i++) {
        while(st.size()>0 && ar[st.top()] >= ar[i]) {
            st.pop();
        }
        if(st.size()>0) {
            ans[i] = st.top();
        }
        else {
            ans[i] = -1
        }
        st.push(i)
    }
    return ans;
}
```

Nearest Greater on Left Side:



#Obs: If y is present then x cannot be ans.

Ex:	0	1	2	3	4	5	6	7
ar[8] =	11	10	6	5	10	7	4	8
ans[] =	-1	11	10	6				



#Idea:

```
while( st.top() <= ar[i]){
    st.pop()
}
```

```
vector<int> nearestElement(vector<int> &ar){
    stack<int> st;
    int n=ar.size();
    vector<int> ans(n,-1);
    for(int i=0; i<n; i++){
        while( st.size()>0 && st.top()<=ar[i]){
            st.pop();
        }
        if(st.size()>0){
            ans[i]= st.top();
        }
        else{
            ans[i]=-1
        }
        st.push(ar[i])
    }
    return ans;
}
```

Next greater index in left

Ex:

	0	1	2	3	4	5	6	7
ar[8] =	11	10	6	5	10	7	4	8

ans[] =

```
vector<int> nearestElement(vector<int> &ar){
    stack<int> st;
    int n=ar.size();
    vector<int> ans(n,-1);
    for(int i=0; i<n; i++){
        while(st.size()>0 && ar[st.top()]<=ar[i]){
            st.pop();
        }
        if(st.size()>0){
            ans[i]=st.top();
        }
        else{
            ans[i]=-1
        }
        st.push(i)
    }
    return ans;
}
```

All Versions: