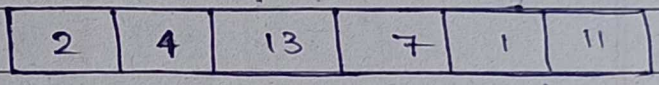
Page No.:
Date: / /

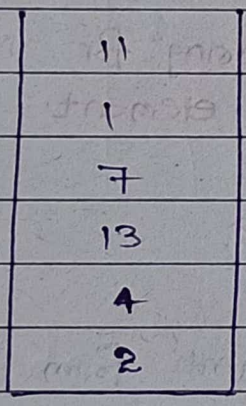
Lecture - 36
(Stack - Medium)

Problem # 1 . Get Minimum element from stack

You have N element
↳ find the min-m element while implementing a stack.

Required time Complexity : O(1)





push ✓
pop ✓
get_min()

Ans = 1

We check one by one all the element and find min-m
but time Complexity is O(N)
and required is O(1)

This approach fails.

We can solve this by Another Approach.

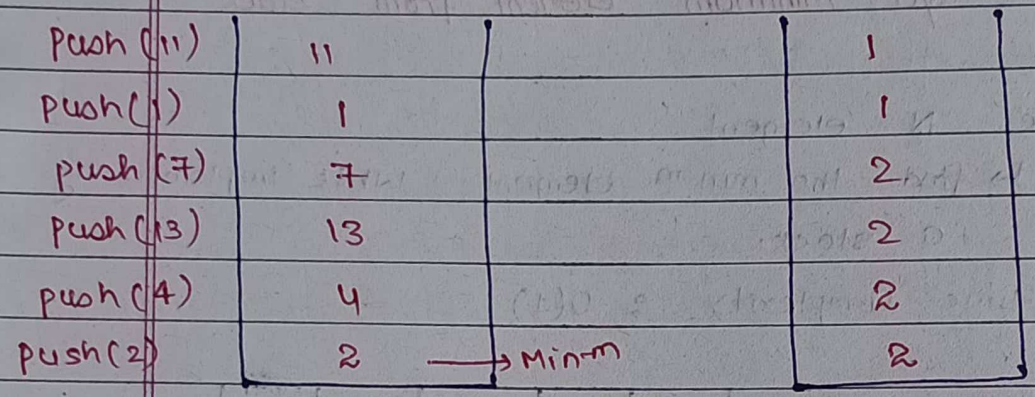
Teacher's Signature.....

Page No:
Date: / /

Approach 2:

2	4	13	7	1	11

Make 2 stack



↳ This stack is storing for Minm element.

We will push all the element in stack in constant time.

We will return the top element from 2nd stack which is minimum.

But here we can take extra space for second stack.

Space Complexity : O(N)

Not Required →

We want time Complexity and space Complexity both as O(1).

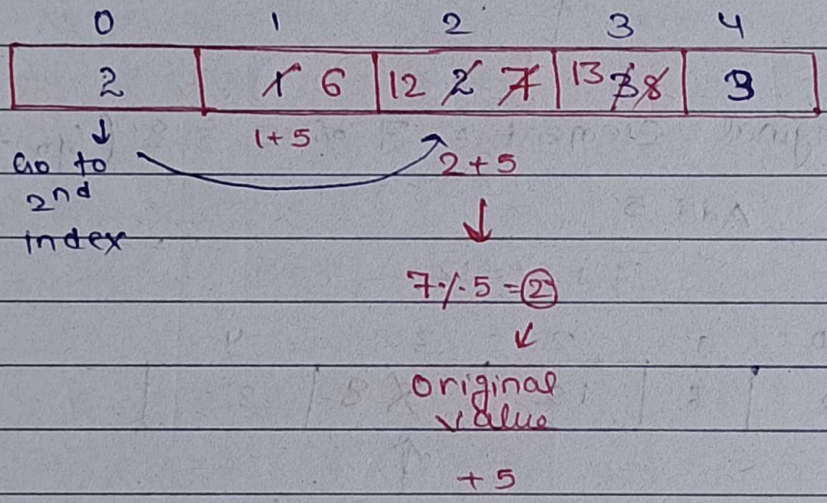
Teacher's Signature.....

Approach 3:

$$\frac{\text{freq} * \text{size} + \text{original\_value}}{\text{size}} = \text{value}$$

frequency = divide

original value = Mod

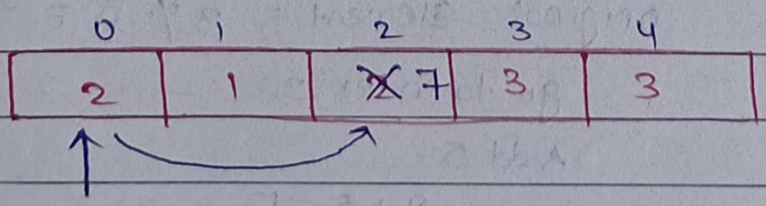


Explanation

Limit of Array element → (0 to 4)

We take a higher value.

Let we take 5.



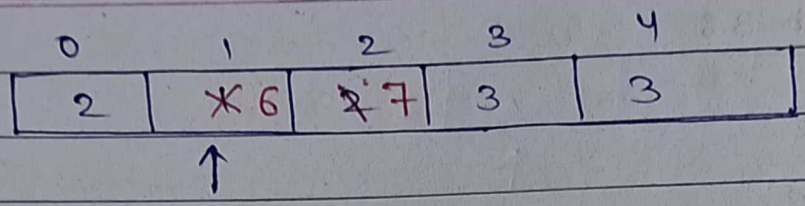
Element = 2 % 5 = 2

Go to index 2

Add 5 to that index

2 + 5 = 7

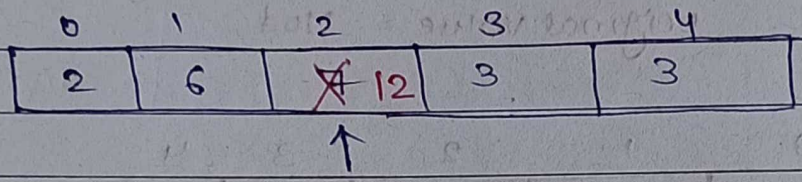
Teacher's Signature.....



Element = 1 % 5 (Go to index 1)

Add 5

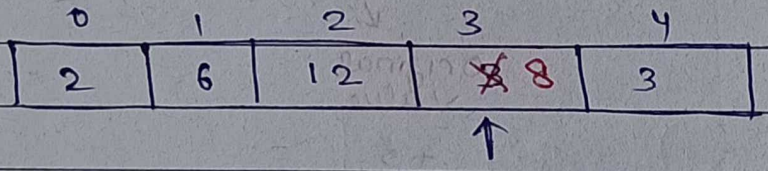
1 + 5 = 6



original element = 7 % 5 = 2 (Go to index 2)

Add 5

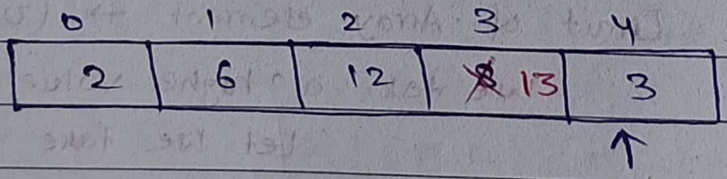
7 + 5 = 12



original element = 3 % 5 (Go to index 3)

Add 5

5 + 3 = 8



original element = 3 % 5 = 3

go to index 3

Add 5

8 + 5 = 13

Teacher's Signature.....

For Finding Frequency, divide all element by 5

Frequency =	0	1	2	2	0

$2/5 = 0$ $6/5 = 1$ $12/5 = 2$ $13/5 = 2$

$3/5 = 0$.

* same Concept we use in stack.

Range of Number is 1 to 100.

Take a number bigger than Range.

Let n = 101

originalvalue + minelement * 101

101

for this = mod For this = divide.

1 <= num <= 100

Why we take 101, why not 102, ... ;

↳ because when we mod we get 0 to 100 which is limit of num.

[Original value + minElem * 101] = value

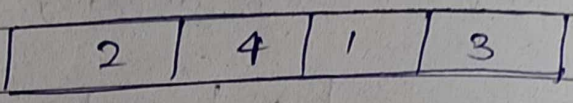
↓

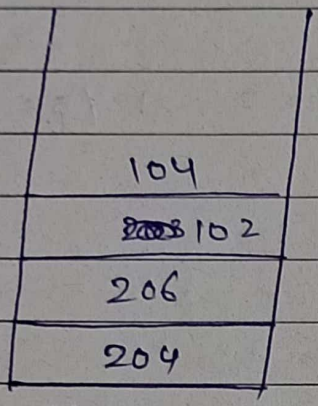
put in stack

value % 101 = original_value

value / 101 = min_element.

Teacher's Signature.....



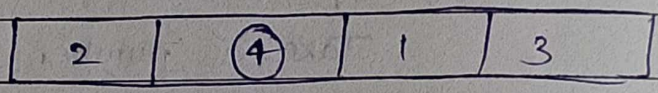


~~value = 2 + 100~~

value = 2 + 101 * (2)

= 204

because till now no min-m present.



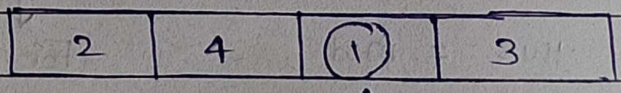
(2, 4) → min = 2

min-m value = 204 / 101

= (2)

Val = 4 + 101 * 2

= 206



min-m element

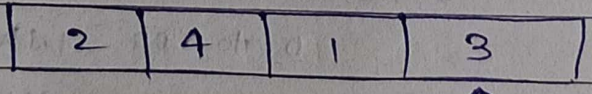
= 206 / 101

= (2)

(2, 1) → min = 1

val = 1 + 101 × 1

= ~~203~~ 102



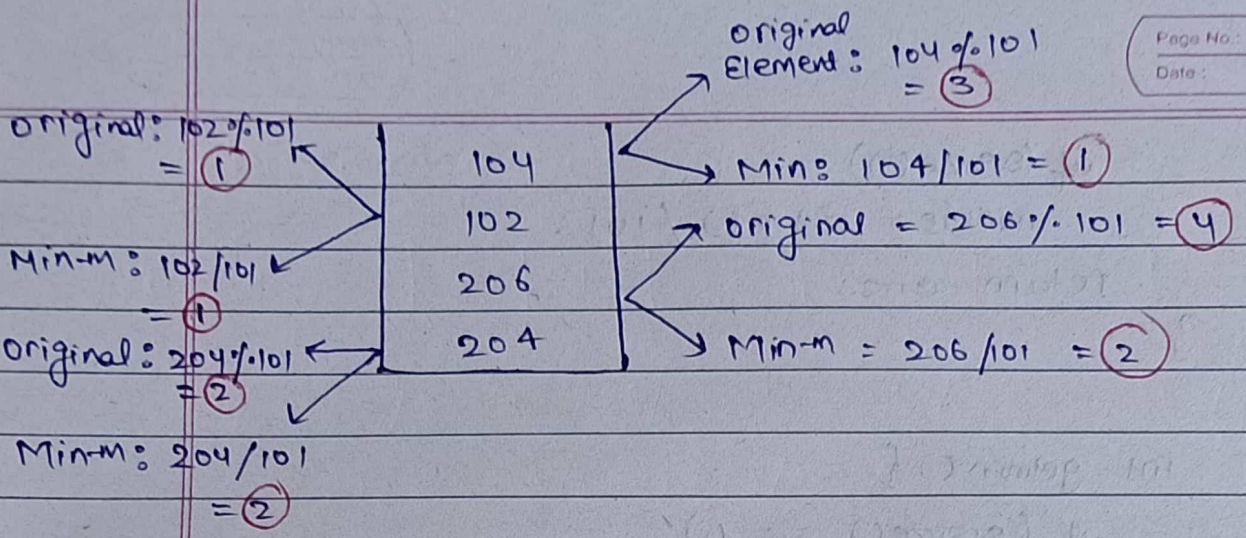
min = 102 / 101

= 1

(1, 3) → min = 1

Val = 3 + 101 × 1

= 104

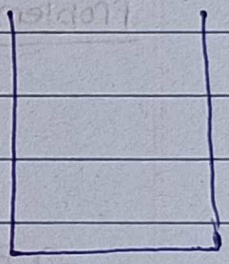


By this We can get both by one stack.

→ original element

→ Min-m Number.

* Implementation of push:



```
Void push (int x) {
    if (s.size() == 0) {
        minEle = x;
        S.push (x + (101 * x));
    }
    else {
        minEle = min (x, S.top() / 101);
        S.push (x + (101 * minEle));
    }
}
```

* Implementation of pop:

```
int pop() {
    if (S.size() == 0)
        return -1;
    int ans = S.top();
    ans = ans % 101;
    S.pop();
```

Teacher's Signature.....

```
    if (s.size())
        minEle = s.top() / 101;
    return ans;
}

int getmin() {
    if (s.size() == 0)
        return -1;
    else
        return s.top() / 101;
}
```

Problem 2: String Manipulation

Sequence of n words.

If two same word come together then he destroy both.

After all destruction happen, how many numbers of words left.

V[] = { ab, aa, aa, bcd, ab }

aa, aa → destroyed

output = 3

Teacher's Signature.....

* v = [tom, jerry, jerry, tom]

destroy

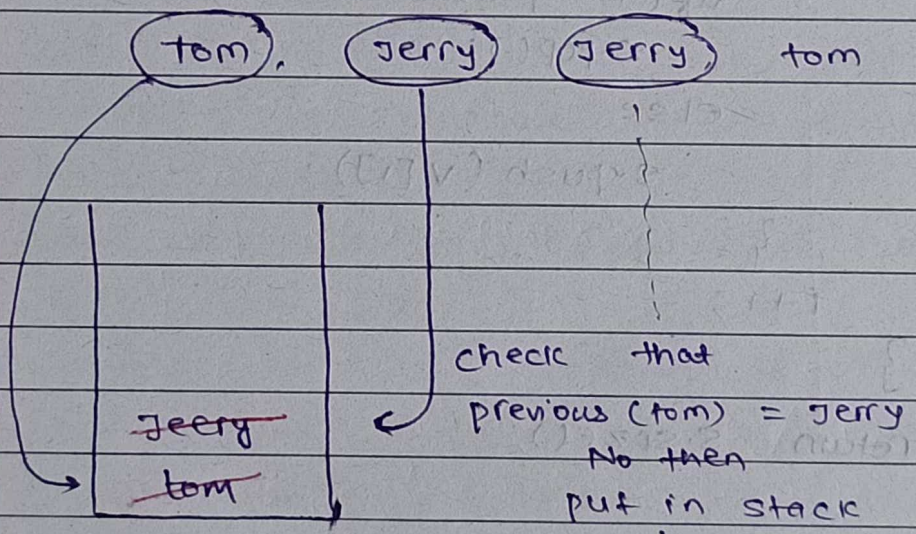
v = [tom, tom]

destroy

v = [ ]

output = 0

* How to solve :



Again jerry

Check stack.top() == jerry

yes

then delete jerry

pop (top)

Again tom, check

stack.top() == tom

Yes

pop (top).

Teacher's Signature.....

then, Count the number of element present in stack and return.

Code:

```
Stack <string> s;
  int i= 0;
  While( i< v.size()) {
      if ( s.size() == 0)
          s.push (v[i]);
      else {
          if ( s.top() == v[i])
              s.pop();
          else
              s.push (v[i]);
        }
      i++;
  }
  return  s.size();
```

Teacher's Signature.....

Page No.
Date : / /

Problem : 3 Make the array beautiful

Given a Array → With negative & non-negative integers.

Condition for beautiful :

two adjacent integer,

arr[i] and arr[i+1] are either Negative or non-negative.

You have to do some operation until your array becomes beautiful :

If two adjacent integer are different i.e. one of them is negative and other is non-negative, remove them.

Empty array is also a beautiful array.

Example

destroyed

Input : 4 (2 -2) 1

output : 4 1

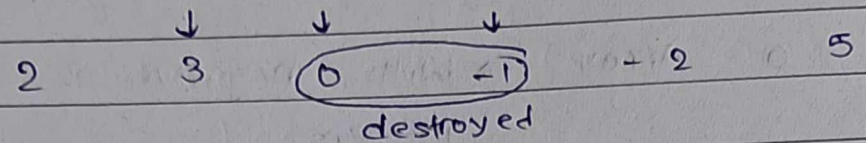
Example :

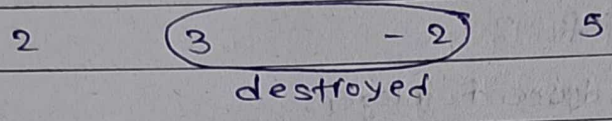
Input : (2 -2) (1 -1)

destroyed destroyed.

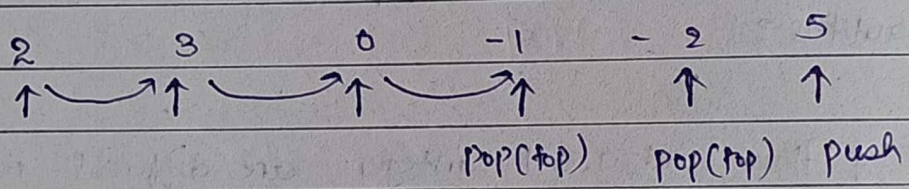
Teacher's Signature.....

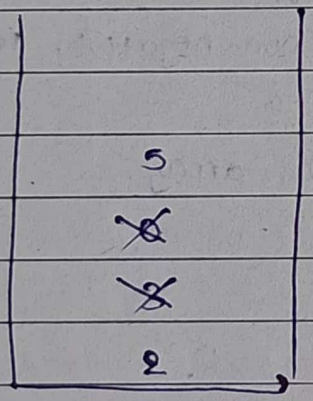
By stack :



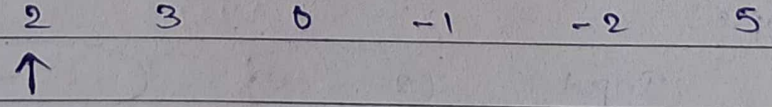


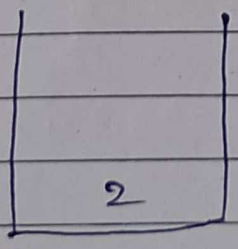
[2, 5] → answer.





detailed explanation :





2 . (stack empty)

Push (2)

Teacher's Signature.....

2 3 0 -1 -2 5

↑

+ve

3
2

top = 2 (+ve)

push(3)

2 3 0 -1 -2 5

↑

0
3
2

0 +ve

top = 3 (+ve)

push

2 3 0 -1 -2 5

↑

~~x~~
3
2

-1 -ve

top = 0 +ve

pop(top)

2 3 0 -1 -2 5

↑

~~x~~
2

-2 (-ve)

top = 3 +ve

pop(top)

2 3 0 -1 -2 5

↑

5
2

5 (+ve)

top = 2 +ve

push

Now, return the element which are present in stack.

Teacher's Signature.....

Page No:
Date: / /

Code:

```
Vector <int> makeBeautiful (vector <int> arr) {
        Stack <int> s ;
    for (int i=0;  i< arr.size();  i++) {
            if ( s .empty ()) {
                s.push (arr[i]);
              }
            else {
             if ((s.top() >= 0 && arr[i] <0) ||
                 (s.top() < 0 && arr[i] >=0 )) {
                    s.pop();
                }
             else {
                 s.push (arr[i]);
                }
          }
    }

vector <int> beautifullArray;
    while (! st.empty()) {
        beautifullArray.push_back (s.top());
            s.pop ();
      }

reverse (beautifullArray.begin (), beautifullArray.end());

return beautifullArray;

}
```

Teacher's Signature.....

Problem :- 4

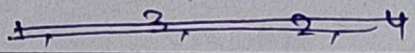
* Next Greater Element :

Given an Array :

return an Array with next greater Element.

Ex : arr[] = [1, 3, 2, 4]

ans = [



~~for 1~~

1, 3, 2, 4

for (1) → Next greater = 3

ans = { 3, }.

for (3) → Next greater = 4

ans = { 3, 4 }

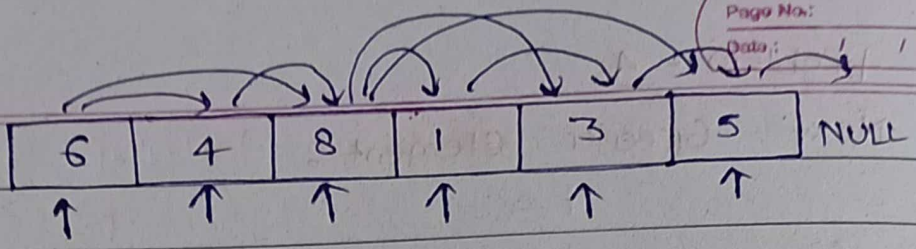
for (2) → Next greater = 4

ans = { 3, 4, 4, }

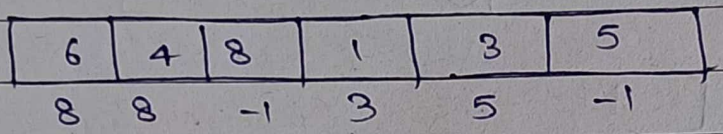
for (4) → Next greater = Not present (-1)

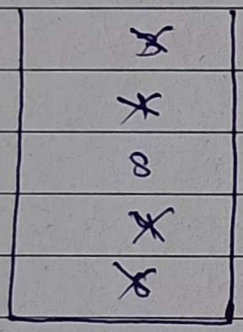
ans = { 3, 4, 4, -1 }.

Teacher's Signature.....



Output : | 8 | 8 | -1 | 3 | 5 | -1 |

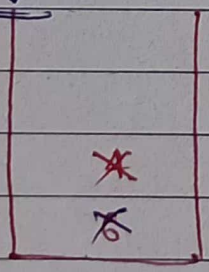




5 → 6 → put 1st element
4 → 4 < 6 put in stack
8 → 8 > 4 , 8 > 6
1 → 1 < 8 put in stack
3 → 3 > 1
5 → 5 > 3

* 6 4 8 3 1 2 5

Next greater 8 8 -1 5 2 5 -1



6 → put in stack

4 → 4 < 6 (then put in stack)

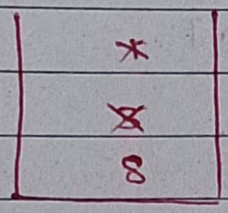
8 → 8 > 4 (then for 4 '8' is next greater

Pop(4)

Teacher's Signature.....

8 > 6 then

for 6 '8' is next greater



pop (top);

3 → 3 < 8 push

1 → 1 < 3 push

2 → 2 > 1 for 1

2 is Next greater

5 → 5 > 2

then for 2

(5 is greater.)

(iii)

for rest (-1) is next greater

Code:

```
stack<int> s;
int i = 0;
while(i<n) {
 if (s.size() == 0)   // stack is empty
    s.push(i);
 else{
    if (arr[s.top()] >= arr[i])
      s.push(i);
    else {
```

Teacher's Signature.....

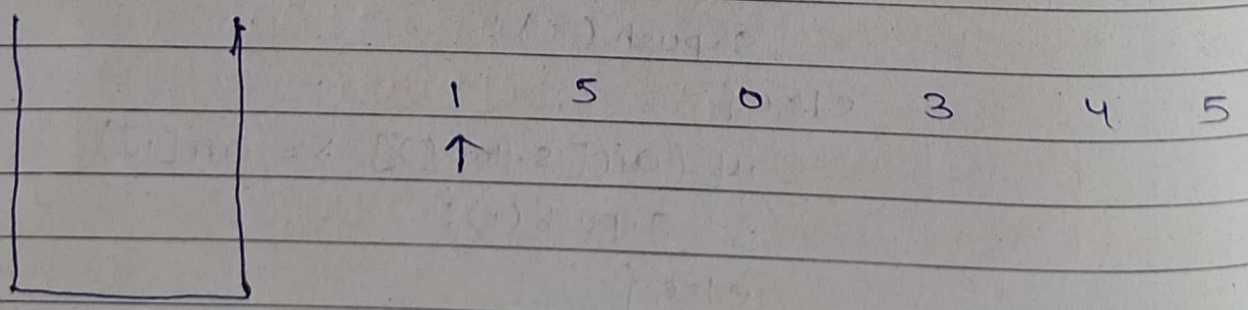
```
        while(s.size() && arr[s.top()] < arr[i])
        {
            ans[s.top()] = arr[i];
            s.pop();
        };
        s.push(i);
      }
    }
    i++;
  }

  while( s.size()) {
      ans[s.top()] = -1;
      s.pop();
    };
    return ans;
  }
}
```

* Problem - 5 : Smallest number on left :

a = [1, 5, 0, 3, 4, 5]

Output : -1 1 -1 0 3 4



(1) → Stack is empty
then store (-1)

5
1

Output = [-1,]

1 5 0 3 4 5
↑

(5) → stack is not empty.
if smaller than arr[i] found
then return.
1 is smaller. and then

output = [-1, 1

1 5 0 3 4 5
↑

0
5
1

(0) → stack is not Empty
check, stack if smaller than arr[i] then return.
otherwise, return -1
5 > 0
1 > 0
return -1

output = [-1, 1, -1

(3) stack is not empty
check smaller in stack
found 0 return 0.

3
0
5
1

output = [-1, 1, -1, 0]

Teacher's Signature.....

4
3
0
5
1

④ Stack is not empty
check smaller than 4
return 3

output = [-1, 1, -1, 0, 3,

⑤ stack is not empty
check smaller than 5
return 4

Output = [-1, 1, -1, 0, 3, 4]

Loop Ends
Return output.

Code :

```
Vector <int> leftsmaller (int n, int a[]){
        Vector <int> result (n, -1);
      stack <int> s;

    for( int i=0; i< n; i++){
        While ( ! s.empty() && s.top() >= a[i]){
                s.pop();
        }
        if ( ! s.empty() ) {
            result[i] = s.top();
        }
        s.push(a[i]);
    }
    return result;
}
```

Teacher's Signature.....