17) LARGEST RECTANGLE IN HISTOGRAM

METHOD:
LINK OF EXPLANATION: Largest rectangle in Histogram | Leetcode #84
This sum is solved efficiently in O(N) time by using the stacks.
In this sum, first of all we should remember that we push the indices into the stack.We also assume 2 arrays for storing the left and right reach from the particular bar.
We push the first element into the stack and the corresponding left reach will be zero for the first element.Then we start iterating the array,

- We go on popping the indices from the stack unless the stack becomes empty or we find the index whose value is smaller than the current value
- If the stack is empty then we didn't get any index whose value is smaller than the current bar. So the left reach for that element will be its own index.
- If the stack is not empty means the index at the stack top has the value smaller than the current bar value, so the reach will be (i-s.top())

We follow the same procedure for finding the right reach for the bars.
And then we find the maximum value among arr[i]*(leftreach +rightreach+1)

CODE OF THE PROGRAM:

```
class Solution
{
  public:
  //Function to find largest rectangular area possible in a given histogram.
  long long getMaxArea(long long arr[], int n)
  {
    stack<int> s;
    int a[n],b[n];
    a[0]=0;
    s.push(0);
    for(int i=1;i<n;i++){
      while(s.size()!=0 && arr[s.top()]>=arr[i]){
        s.pop();
      }
      if(s.size()==0){
        a[i]=i;
      }
      else{
        a[i]=i-s.top()-1;
      }
      s.push(i);
    }
    while(s.size()!=0){
      s.pop();
    }
    b[n-1]=0;
    s.push(n-1);
    for(int i=n-2;i>=0;i--){
      while(s.size()!=0 && arr[s.top()]>=arr[i]){
```

```
            s.pop();
        }
        if(s.size()==0){
            b[i]=n-1-i;
        }
        else{
            b[i]=s.top()-i-1;
        }
        s.push(i);
    }
    long long int maximum=0;
    for(int i=0;i<n;i++){
        maximum=max(maximum,arr[i]*(a[i]+b[i]+1));
    }
    return maximum;
  }
};
```