STACK

Q1)Array implementation on stack

By array:

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
 struct MyStack{
   int *arr;
   int cap;
   int top;
   //intialising
   MyStack(int c){//constructor
      cap=c;
      arr=new int [cap];
      top=-1;
   }
   void push(int x){
      if(top==cap-1){
         cout<<"Stack is full"<<endl;
         return;
      }
      top++;
      arr[top]=x;
   }

   int pop(){
      if(top==-1){
         cout<<"Stack is Empty"<<endl;
         return INT_MIN;
      }
      int res=arr[top];
      top--;
      return res;
   }
   int peek(){
      if(top==-1){
         cout<<"Stack is Empty"<<endl;
         return INT_MIN;
      }
      return arr[top];
   }

   int size(){
      return (top+1);
   }
   bool isEmpty(){
      return top==-1;
   }
};


int main(){
   //remember the size==5
   MyStack s(5);
   s.push(5);
   s.push(10);
   s.push(15);
   cout<<s.pop()<<endl;
   cout<<s.size()<<endl;
   cout<<s.peek()<<endl;
   cout<<s.isEmpty()<<endl;
   return 0;
}
```

By vector:

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
 struct MyStack{
   vector<int> v;
   void push(int x){
      v.push_back(x);
   }
   int pop(){
      int res=v.back();
      v.pop_back();
      return res;
   }
   int peek(){
      return v.back();
   }
   int size(){
      return v.size();
   }
   bool isEmpty(){
      return v.empty();
   }
};
int main(){
   //here the size isnt specified
   MyStack s;
   s.push(5);
```

```
   s.push(10);
   s.push(15);
   cout<<s.pop()<<endl;
   cout<<s.size()<<endl;
   cout<<s.peek()<<endl;
   cout<<s.isEmpty()<<endl;
}
```

By stack:

```
#include <iostream>
#include <stack>
using namespace std;

int main(){
   stack<int>s;
   int n,m,x;
   cin>>n;
   //input
   for(int i=0;i<n;i++){
      cin>>x;
      s.push(x);
   }
   cout<<s.size()<<endl;
   cout<<s.top()<<endl;
   s.pop();
   cout<<s.top()<<endl;
   s.push(5);
   cout<<s.top()<<endl;
   //for emptying the stack
   while(s.empty()==false){
      cout<<s.top()<<endl;
      s.pop();
   }
   return 0;

}
```

Q2) balanced parenthesis

It's a popular interview problem often displaying the importance of stack.

Code:

```
#include <bits/stdc++.h>
using namespace std;
bool isBalanced(string s) {
   stack<char> st;//remember the char stack
   for (char c : s) {
      if (c == '(' || c == '[' || c == '{') {
         st.push(c);
      } else {
         if (st.empty()) {
            return false;
         } else if (c == ')' && st.top() != '(') {
            return false;
         } else if (c == ']' && st.top() != '[') {
            return false;
         } else if (c == '}' && st.top() != '{') {
            return false;
         } else {
            st.pop();
         }
      }
   }
   return st.empty();
}


int main(){
  string s;
  cin>>s;
   if (isBalanced(s)) {
      cout << "The parentheses in the string are
balanced." << endl;
   } else {
      cout << "The parentheses in the string are
not balanced." << endl;
   }
}
```

Q3)Implement two stacks in an array

```
#include <bits/stdc++.h>
using namespace std;

struct TwoStacks {
   int* arr;
   int cap;
   int top1, top2;

   TwoStacks(int n)   {
      cap = n;
      arr = new int[n];
```

```
    top1 = -1;
    top2 = cap;
}

void push1(int x){
    if (top1 < top2 - 1) {
        top1++;
        arr[top1] = x;
    }
    else {
        cout << "Stack Overflow";
        exit(1);
    }
}

void push2(int x){
    if (top1 < top2 - 1) {
        top2--;
        arr[top2] = x;
    }
    else {
        cout << "Stack Overflow";
        exit(1);
    }
}

int pop1() {
    if (top1 >= 0) {
        int x = arr[top1];
        top1--;
        return x;
    }
    else{
        cout << "Stack UnderFlow";
        exit(1);
    }
}

int pop2()
{
    if (top2 < cap) {
        int x = arr[top2];
        top2++;
        return x;
    }
    else {
        cout << "Stack UnderFlow";
        exit(1);
    }
}
};

int main()
{
    TwoStacks ts(5);
    ts.push1(5);
    ts.push2(10);
    ts.push2(15);
    ts.push1(11);
    ts.push2(7);
    cout << "Popped element from stack1 is
"<<ts.pop1();
    ts.push2(40);
    cout << "\nPopped element from stack2 is
"<< ts.pop2();
    return 0;
}
```

Q4)Implement k stacks in an array

```
#include <bits/stdc++.h>
using namespace std;

struct kStacks {
    int *arr;
    int *top;
    int *next;
    int cap, k;
    int freeTop;
    //intialisation step
    //top=-1 and next=1,2,4 and last of it -1
    //free top=0
    kStacks(int k1, int n){
        k = k1; cap = n;
        arr = new int[cap];
        top = new int[k];
        next = new int[cap];
        for (int i = 0; i < k; i++){
            top[i] = -1;
        }
        freeTop = 0;
        for (int i=0; i<cap-1; i++){
            next[i] = i+1;
        }
        next[cap-1] = -1;
    }
```

```
    bool isFull(){
        return (freeTop == -1);
        }
    bool isEmpty(int sn){
        return (top[sn] == -1);
    }
    //full function
    void push(int x, int sn) {
    if (isFull()) {
        cout << "\nStack Overflow\n";
        return;
    }
    int i = freeTop;
    freeTop = next[i];
    next[i] = top[sn];
    top[sn] = i;
    arr[i] = x;
    }
    //pop function
    int pop(int sn) {
    if (isEmpty(sn)) {
        cout << "\nStack Underflow\n";
        return INT_MAX;
    }
    int i = top[sn];
    top[sn] = next[i];
    next[i] = freeTop;
    freeTop = i;
    return arr[i];
    }
};
int main()
{
    int k = 3, n = 10;
    kStacks ks(k, n);


    ks.push(15, 2);
    ks.push(45, 2);

    ks.push(17, 1);
    ks.push(49, 1);
    ks.push(39, 1);

    ks.push(11, 0);
    ks.push(9, 0);
    ks.push(7, 0);

    cout << "Popped element from stack 2 is "
<< ks.pop(2) << endl;
    cout << "Popped element from stack 1 is "
<< ks.pop(1) << endl;
    cout << "Popped element from stack 0 is "
<< ks.pop(0) << endl;

    return 0;
}
```

Q5)Stock span problem

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void solve(vector<ll>&v){
    ll n=v.size();
    stack<ll>s;
    s.push(0);
    cout<<1<<" ";
    for(ll i=1;i<n;i++){
        while(s.empty()==false &&
v[i]>=v[s.top()]){
            s.pop();
        }
        int span=s.empty()?i+1:i-s.top();
        cout<<span<<" ";
        s.push(i);
    }
}
int main(){
    vector<ll>v;
    ll n,m,temp;
    cin>>n;
    for(ll i=0;i<n;i++){
        cin>>temp;
        v.push_back(temp);
    }
    solve(v);

}
```

Q6)previous greater element

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void solve(vector<ll>&v){
    ll n=v.size();
    stack<ll>s;
    s.push(v[0]);
```

```
    cout<<-1<<" ";
    for(ll i=1;i<n;i++){
        while(s.empty()==false && v[i]>=s.top()){
            s.pop();
        }
        int span=s.empty()?-1:s.top();
        cout<<span<<" ";
        s.push(v[i]);
    }
}
int main(){
    vector<ll>v;
    ll n,m,temp;
    cin>>n;
    for(ll i=0;i<n;i++){
        cin>>temp;
        v.push_back(temp);
    }
    solve(v);

}
```

Q7)Next greater element

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void solve(vector<ll>&v){
    ll n=v.size();
    stack<ll>s;
     vector<ll>v2;
    reverse(v.begin(),v.end());
    s.push(v[0]);
    cout<<-1<<" ";
    for(ll i=1;i<n;i++){
        while(s.empty()==false && v[i]>=s.top()){
            s.pop();
        }
        int span=s.empty()?-1:s.top();

        v2.push_back(span);
        s.push(v[i]);
    }
    reverse(v2.begin(),v2.end());
    for(auto x:v2){
        cout<<x<<" ";
    }
}
int main(){
    vector<ll>v;
    ll n,m,temp;
    cin>>n;
    for(ll i=0;i<n;i++){
        cin>>temp;
        v.push_back(temp);
    }
    solve(v);

}
```

Q8)largest rectangular area

Naïve:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int solve(vector<ll>&arr){
    ll n=arr.size();
   int res=0;
    int ps[n],ns[n];

    stack <int> s;
    s.push(0);
    for(int i=0;i<n;i++){
        while(s.empty()==false &&
arr[s.top()]>=arr[i])
            s.pop();
        int pse=s.empty()?-1:s.top();
        ps[i]=pse;
        s.push(i);
    }

    while(s.empty()==false){
        s.pop();
    }

    s.push(n-1);
    for(int i=n-1;i>0;i--){
        while(s.empty()==false &&
arr[s.top()]>=arr[i])
            s.pop();
        int nse=s.empty()?n:s.top();
        ns[i]=nse;
        s.push(i);
    }

    for(int i=0;i<n;i++){
        int curr=arr[i];
        curr+=(i-ps[i]-1)*arr[i];
        curr+=(ns[i]-i-1)*arr[i];
```

```
    res=max(res,curr);
  }
  return res;
}
int main(){
  vector<ll>v;
  ll n,m,temp;
  cin>>n;
  for(ll i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
  }
  cout<<solve(v);

}
```

Pro:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int solve(vector<ll>&arr){
  ll n=arr.size();
  stack <int> s;
  int res=0;
  int tp;
  int curr;

  for(int i=0;i<n;i++){
    while(s.empty()==false &&
arr[s.top()]>=arr[i]){
      tp=s.top();s.pop();
      curr=arr[tp]* (s.empty() ? i : i - s.top() -
1);
      res=max(res,curr);
    }
    s.push(i);
  }
  while(s.empty()==false){
    tp=s.top();s.pop();
    curr=arr[tp]* (s.empty() ? n : n - s.top() -
1);
    res=max(res,curr);
  }

  return res;
}
int main(){
  vector<ll>v;
  ll n,m,temp;
  cin>>n;
  for(ll i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
  }
  cout<<solve(v);

}
```

Q9)Largest rectangle of all 1s

This solution is quite similar to the previous question with a observable pattern:

```
#include <bits/stdc++.h>
using namespace std;
#define R 4
#define C 4
int maxHist(int row[]){
  stack<int> result;
  int top_val;
  int max_area = 0;
  int area = 0;
  int i = 0;
  while (i < C) {
    if (result.empty() || row[result.top()] <=
row[i])
      result.push(i++);
    else {
      top_val = row[result.top()];
      result.pop();
      area = top_val * i;
      if (!result.empty())
        area = top_val * (i - result.top() - 1);
      max_area = max(area, max_area);
    }
  }
  while (!result.empty()) {
    top_val = row[result.top()];
    result.pop();
    area = top_val * i;
    if (!result.empty())
      area = top_val * (i - result.top() - 1);

    max_area = max(area, max_area);
  }
  return max_area;
}

int maxRectangle(int A[][C]){
```

```
    int result = maxHist(A[0]);
    for (int i = 1; i < R; i++) {
        for (int j = 0; j < C; j++)
            if (A[i][j])
                A[i][j] += A[i - 1][j];

        result = max(result, maxHist(A[i]));
    }

    return result;
}

int main()
{
    int A[][C] = {
        { 0, 1, 1, 0 },
        { 1, 1, 1, 1 },
        { 1, 1, 1, 1 },
        { 1, 1, 0, 0 },
    };

    cout << "Area of maximum rectangle is "
         << maxRectangle(A);

    return 0;
}
```

Q10)design Stack with min O(1) space

Naïve: (handles only positive cases)

```
#include <bits/stdc++.h>
using namespace std;

struct MyStack {

    stack<int> s;
    int min;

void push(int x) {

     if(s.empty() ) {
        min=x;
        s.push(x);
     }
     else if(x<=min){
         s.push(x-min);
         min=x;
     }else{
     s.push(x);
     }
  }

int pop() {

    int t=s.top();s.pop();
    if(t<=0){
        int res=min;
        min=min-t;
        return res;
    }else{
        return t;
    }
  }

int top() {
    int t=s.top();
    return ((t<=0)? min : t);
  }

int getMin() {
     return min;
  }
};

int main()
{
    MyStack s;
    s.push(4);
    s.push(5);
    s.push(8);
    s.push(1);
    s.pop();

    cout<<" Minimum Element from Stack: "
<<s.getMin();
    return 0;
}
```

Pro: (handles both positive as well as negative cases)p

https://leetcode.com/problems/min-stack/

```
class MinStack {
public:
stack<long long int>s;
long long int min;
    MinStack() {

    }
```

```
    void push(long long int val) {
        if(s.empty()){
            min=val;
            s.push(val);
        }else if(val<min){
            s.push((2*val)-min);
            min=val;
        }else{
           s.push(val);
        }
    }

    void pop() {
         if(s.top()>=min){
            s.pop();
        }else if(s.top()<min){
            min = (2*min)-s.top();
            s.pop();
        }
    }

    int top() {
        if(s.empty()){
            return -1;
        }else {
            return (s.top()>=min)?s.top():min;
        }
    }

    int getMin() {
        return s.empty()?0:min;
    }
};
```

Q11)Design a infix to postfix conversion

Q12)https://leetcode.com/problems/asteroid-collision/description/

Naïve:

```
class Solution {
public:
    vector<int> asteroidCollision(vector<int>&
asteroids) {
        vector<int> v;
        stack<int> s;
        long long int n = asteroids.size();

        for (int i = 0; i < n; i++) {
            if (asteroids[i] > 0) {
                s.push(asteroids[i]);
            } else if (asteroids[i] < 0) {
                long long int m = abs(asteroids[i]);
                while (!s.empty() && s.top() < m) {
                    s.pop();
                }
                if (!s.empty() && s.top() == m) {
                    s.pop();
                }
            }
        }

        while (!s.empty()) {
            v.push_back(s.top());
            s.pop();
        }

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

        return v;
    }
};
```

Pro:

(my method stack)—by brute force)

Passed-241/270

Note—this doesn't handle {-2,1,-2,-2} case

```
class Solution {
public:
    vector<int> asteroidCollision(vector<int>&
asteroids) {
        vector<int> v;
        stack<int> s;
        stack<int>s1;
        stack<int>s3;
        stack<int>s4;
        long long int n = asteroids.size();
        long long int k=0;
        bool flag1=true;
        for(int i=0;i<n;i++){
            if(asteroids[i]>0){
                break;
            }else{
                s1.push(asteroids[i]);
                k++;
                flag1=false;
            }
        }
        int prev;
        for (int i = k; i < n; i++) {
```

```
        if (asteroids[i] > 0) {
            s.push(asteroids[i]);
        } else if (asteroids[i] < 0) {
            long long int m = abs(asteroids[i]);
            while (!s.empty() && s.top() < m) {
                s.pop();
            }
            if(s.empty()==true){
                s.push(asteroids[i]);
            }
            if (!s.empty() && s.top() == m) {
                s.pop();
            }
            prev=asteroids[i];
        }
    }
    s4=s;
    if(flag1==false){
    while(!s4.empty()){
      s3.push(s4.top());
      s4.pop();
    }
    while(!s3.empty()){
       s1.push(s3.top());
       s3.pop();
    }
     while (!s1.empty()) {
        v.push_back(s1.top());
        s1.pop();
    }
    }
    if(flag1==true){
    while (!s.empty()) {
        v.push_back(s.top());
        s.pop();
    }
    }

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

    return v;
  }
};
```

Best way:

```
class Solution {
public:
  vector<int> asteroidCollision(vector<int>&
asteroids) {
    stack<int> st; // Stack of integer...
    for(auto &ast : asteroids){
      bool destroy = false; // Initially, NO
COLLISION...
      if(ast > 0){
        st.push(ast); // +ve k case mein...
      }
      else{ // -ve k case jb collision nhi hua h...
        if(st.empty() || st.top() < 0){
          st.push(ast);
        }
        else{ // COLLISION...
          while(!st.empty() && st.top() > 0){
            if(abs(ast) == st.top()){
              destroy = true;
              st.pop();
              break;
            }
            else if(abs(ast) > st.top()){
              st.pop();
            }
            else{
              destroy = true;
              break;
            }
          }
          if(!destroy){
            st.push(ast);
          }
        }
      }
    }
    vector<int> ans(st.size());
    // REVERSE...
    for(int i = st.size() - 1; i >= 0; --i){
      ans[i] = st.top();
      st.pop();
    }
    return ans;
  }
};
```

Q13) https://leetcode.com/problems/sum-of-subarray-minimums/

```
class Solution {
public:
   vector<int> getNSL(vector<int>& arr, int n)
{
      vector<int> result(n);
      stack<int> st;

      for(int i = 0; i<n; i++) {
        if(st.empty()) {
          result[i] = -1;
        } else {
```

```
            while(!st.empty() && arr[st.top()] > arr[i]) //strictly less
                st.pop();

            result[i] = st.empty() ? -1 : st.top();
        }
        st.push(i);
    }

    return result;
  }

  //This is just we are finding next smaller to each element to right
  //Similar : Leetcode-84
  vector<int> getNSR(vector<int>& arr, int n) {
    vector<int> result(n);
    stack<int> st;

    for(int i = n-1; i>=0; i--) {
        if(st.empty()) {
            result[i] = n;
        } else {
            while(!st.empty() && arr[st.top()] >= arr[i]) //non-strictly less
                st.pop();

            result[i] = st.empty() ? n : st.top();
        }
        st.push(i);
    }

    return result;
  }

  int sumSubarrayMins(vector<int>& arr) {
    int n = arr.size();

    vector<int> NSL = getNSL(arr, n); //Next smaller to left
    vector<int> NSR = getNSR(arr, n); //Next smaller to right


    long long sum = 0;
    int M = 1e9+7;
    for(int i = 0; i<n; i++) {

        long long d1 = i - NSL[i]; //distance to nearest smaller to left from i

        long long d2 = NSR[i] - i; //distance to nearest smaller to right from i

        /*
           we have d1 numbers in the left and d2 numbers in the right
           i.e. We have d1 options to start from the left of arr[i]
           and d2 options to end in the right of arr[i]
           so the total options to start and end are d1*d2
        */
        long long total_ways_for_i_min = d1*d2;
        long long sum_i_in_total_ways  = arr[i] * (total_ways_for_i_min);

        sum  = (sum + sum_i_in_total_ways)%M;
      }

      return sum;

   }

};
```

quite similar problem to try:
https://leetcode.com/problems/sum-of-subarray-ranges/description/

https://leetcode.com/problems/longest-valid-parentheses/description/