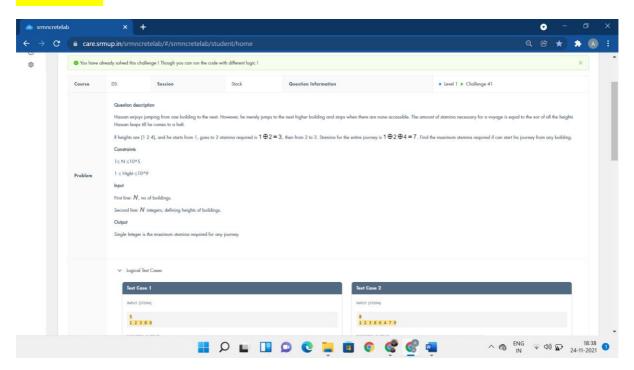
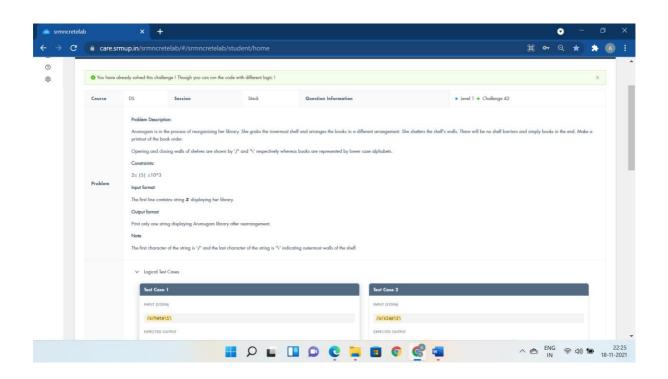
```
int main(){
    node *start = NULL;
    int n,t;
    cin>>n;
    while(n--){
        cin>>t;
        push(&start,t);
    }
    cout<<"Linked List:";
    printList(start);
    return 0;
    cout<<"p1->next=start; void display()";
}
```

## **STACKS:-**

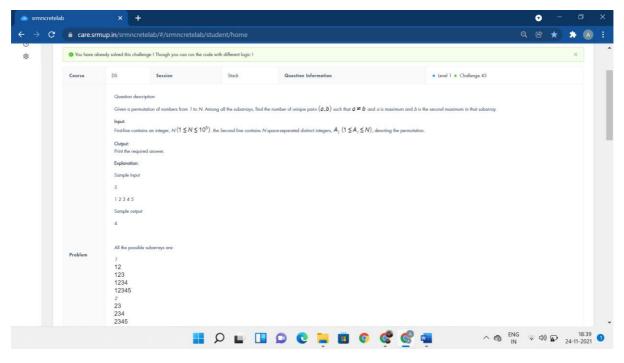


```
#include <stdio.h>
int main() {
  int i, j, arr[1000000], n, temp=0,st[1000000]= {0};
  scanf("%d",&n);
```

```
for(i=0;i< n;i++){}
     scanf("%d",&arr[i]);
  st[n-1] = arr[n-1];
  temp = arr[n-1];
  for(i=n-2;i>=0;i--) {
     for(j=i+1;j< n;j++)
     if(arr[i]<arr[j]) {</pre>
        st[i]=arr[i]^st[j];
     break;
    }
  if(st[i] == 0)
     st[i] = arr[i];
  if(st[i] > temp)
     temp = st[i];
  printf("%d",temp);
  return 0;
}
```



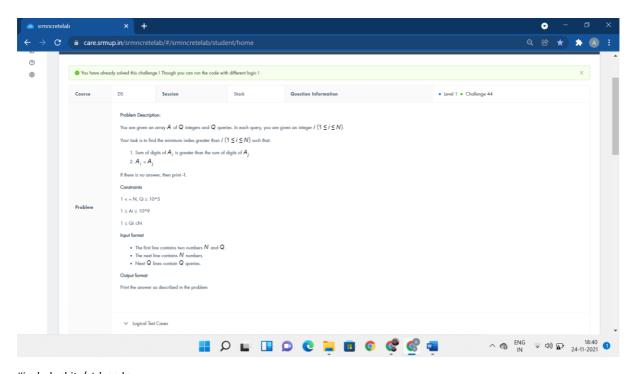
```
#include <bits/stdc++.h>
using namespace std;
int main()
{
  string s,temp="";
  cin>>s;
  stack<string> stk;
  for (unsigned int i = 0; i < s.size(); i++) {
    if(s[i]==47 | |s[i]==92){
      if(!temp.empty()){
         stk.push(temp);
         temp.clear();
      }
    }
    else{
      temp.push_back(s[i]);
    }
  }
  while(!stk.empty()){
    cout<<stk.top();
    stk.pop();
  }
          return 0;
          printf("typedef struct stackvoid arranging(char *s,int n,stack *p)arranging(S,strlen(S),&s1);");
}
```



#include <stdio.h>

}

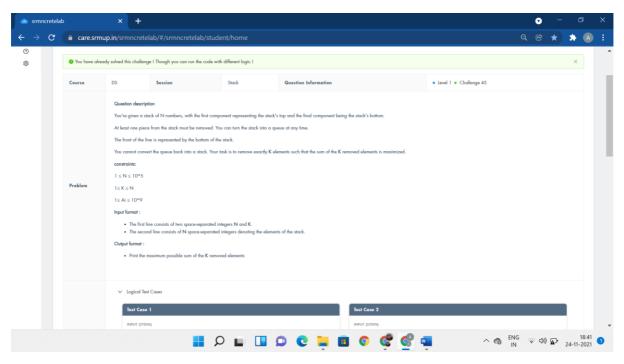
```
int main(){
  int num,i,count=0,a[100001],stck[100001],top=-1;
         scanf("%d", &num);
  for (i=0;i<num;i++) {
                   scanf("%d",&a[i]);
                   while(top!=-1 && stck[top]<a[i]) \{
                             top--;
                             count++;
                   }
                   if (top!=-1) {
                             count++;
                   }
                   stck[++top]=a[i];
         }
          printf("%d",count);
          return 0;
```



```
#include<bits/stdc++.h>
using namespace std;
int main(){
  int n,q;
  cin>>n>>q;
  int *a=new int [n];
  for(int i=0;i< n;i++){
    cin>>a[i];
  int *arr=new int[n];
  for(int i=0;i<n;i++){
    int z=a[i];
    int sum=0;
    while(z>0){
      sum+=(z%10);
      z=z/10;
    }
    arr[i]=sum;
  }
  while(q--){
    int Q;
```

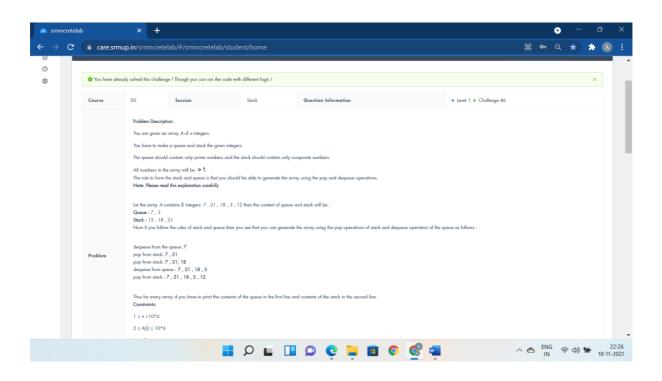
cin>>Q;

```
int ans=-1;
for(int i=Q;i<n;i++){
    if(a[i]>a[Q-1] && arr[i]<arr[Q-1]){
        ans=i+1;
        break;
    }else{
        continue;
    }
}
cout<<ans<<'';
}
return 0;
cout<<"if(arr[x]<arr[y]) if(arr2[x]>arr2[y]) ";
}
```

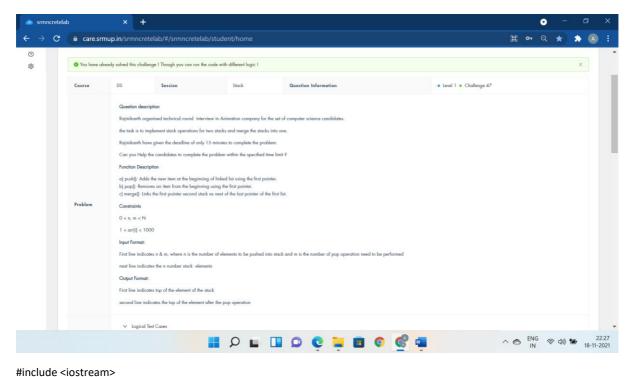


```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   int n,k,i;
   cin>>n>>k;
   int sum = 0;
```

```
int arr[n];
  stack<int>st, st2;
  for(i=0;i<n;i++){
    cin >> arr[i];
    st.push(arr[i]);
  for(i=0;i<k;i++){
    st2.push(arr[i]);
    sum += arr[i];
  }
  int maxs = sum;
  while(k-- > 1){
    sum -= st2.top();
    st2.pop();
    sum += st.top();
    st.pop();
    if(sum > maxs) maxs = sum;
  cout << maxs;
  return 0;
}
```



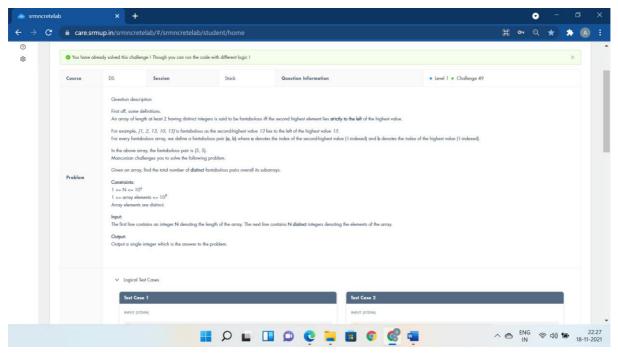
```
#include<bits/stdc++.h>
using namespace std;
bool isPrime(int n)
{
  if(n<=1)
  return false;
  for(int i=2;i<n;i++)
  if(n%i==0)
  return false;
  return true;
}
int main(){
  stack<int> stack;
  int n;
  cin>>n;
  int a[n];
  for(int i=0;i<n;i++){
    cin>>a[i];
    if(isPrime(a[i]))
    cout<<a[i]<<" ";
    else
    stack.push(a[i]);
  }
  cout<<endl;
  while(!stack.empty()){
    cout<<stack.top()<<" ";
    stack.pop();
  }
  return 0;
  cout<<"int read_int() void push(int stack[],int data) top++;";</pre>
}
```



```
using namespace std;
class node {
public:
         int data;
         node* next;
};
class mystack {
public:
         node* head;
         node* tail;
         mystack()
         {
                   head = NULL;
                   tail = NULL;
         }
};
mystack* create()
{
         mystack* ms = new mystack();
         return ms;
```

```
}
void push(int data,mystack* ms)
{
         node* temp = new node();
         temp->data = data;
         temp->next = ms->head;
         if (ms->head == NULL)
                  ms->tail = temp;
         ms->head = temp;
}
int pop(mystack* ms)
{
         if (ms->head == NULL) {
                  cout << "stack underflow" << endl;</pre>
                  return 0;
         }
         else {
                  node* temp = ms->head;
                  ms->head = ms->head->next;
                  int popped = temp->data;
                  delete temp;
                  return popped;
         }
}
void merge(mystack* ms1,mystack* ms2)
{
if (ms1->head == NULL)
{
         ms1->head = ms2->head;
         ms1->tail = ms2->tail;
         return;
}
ms1->tail->next = ms2->head;
```

```
ms1->tail = ms2->tail;
}
void display(mystack* ms)
{
         node* temp = ms->head;
         while (temp != NULL) {
                   cout << temp->data << " ";
                   temp = temp->next;
         }
}
int main()
{
         mystack* ms1 = create();
         mystack* ms2 = create();
         int n,m,t;
         cin>>n>>m;
         for(int i=0;i<n;i++)
           cin>>t;
           push(t,ms1);
         }
         for(int i=0;i< m;i++)
         {
           cin>>t;
           push(t,ms2);
         }
         merge(ms1, ms2);
         for(int i=0;i<n+m;i++)
         cout<<pop(ms1)<<" ";
}
```



#include <bits/stdc++.h>

```
#define sci(x) scanf("%d", &x)
#define scl(x) scanf("%lld", &x)

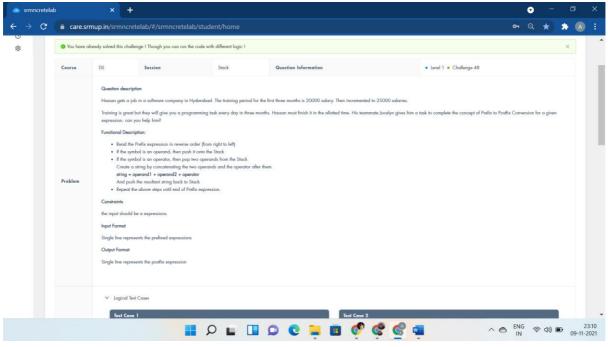
int arr[1000001], cnt[1000001];
int v[1000001];
stack <int> st;

void don(){
    cout<<"void push(llint num)stack[top++]=num;pop()";
}

int main()
{
    int n, i, x;
    sci(n);
    for (i = 1; i <= n; ++i) sci(arr[i]);

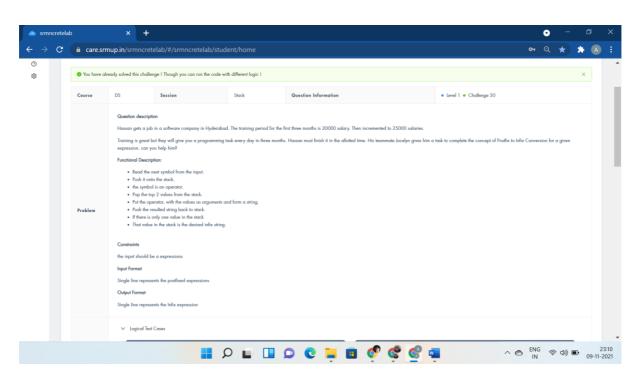
for (i = n; i > 0; --i) {
```

```
while (!st.empty() && arr[i] > arr[st.top()]) {
                                cnt[st.top()] = st.top() - i;
                                st.pop();
                     }
                     st.push(i);
          }
          while (!st.empty()) {
                     cnt[st.top()] = st.top();
                     st.pop();
          }
          for (i = 1; i <= n; ++i) {
                     while (!st.empty() && arr[st.top()] < arr[i]) {
                                x = i - st.top() + 1;
                                v[x] = max(v[x], cnt[st.top()]);
                                st.pop();
                     }
                     st.push(i);
          }
          int k = 0;
          for (i = 2; i \le n; ++i) {
                     k += v[i];
          }
          cout << k << endl;
          return 0;
}
```



#include <iostream> #include <stack> using namespace std; bool isOperator(char x) { switch (x) { case '+': case '-': case '/': case '\*': return true; } return false; } string preToPost(string pre\_exp) { stack<string> s; int length = pre\_exp.size(); for (int i = length - 1;  $i \ge 0$ ; i--) { if (isOperator(pre\_exp[i])) {

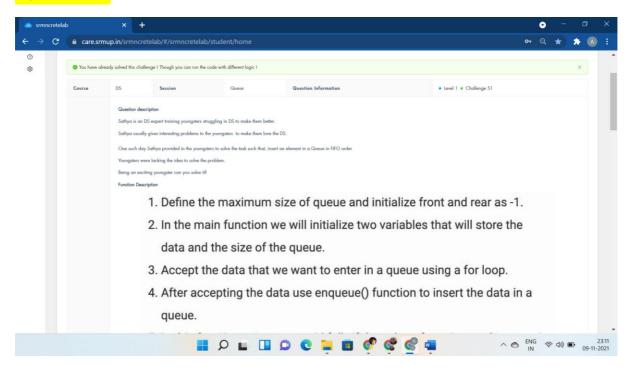
```
string op1 = s.top();
                              s.pop();
                              string op2 = s.top();
                              s.pop();
                              string temp = op1 + op2 + pre_exp[i];
                              s.push(temp);
                    }
                    else {
                              s.push(string(1, pre_exp[i]));
                    }
          }
          return s.top();
}
int main()
{
          string pre_exp;
          cin>>pre_exp;
          cout << "Postfix:" << preToPost(pre_exp);</pre>
          return 0;
}
```



```
#include<iostream>
#include<string.h>
using namespace std;
bool isOperand(char x){
  return (x>='a' && x<='z') || (x >= 'A' && x <= 'Z');
}
string getInfix(string exp)
{
  stack<string> s;
  for(int i=0; exp[i]!='\0'; i++)
    if(isOperand(exp[i]))
      string op(1, exp[i]);
      s.push(op);
    }
    else
    {
      string op1 = s.top();
      s.pop();
      string op2=s.top();
      s.pop();
      s.push("(" + op2 + exp[i] + op1 + ")");
    }
  return(s.top());
}
int main()
  string exp;
  cin>>exp;
```

```
cout<<getInfix(exp);
return 0;
}</pre>
```

## **QUEUES:-**



```
#include <stdio.h>
#define SIZE 100
void enqueue(int);
void display();
int items[SIZE], front = -1, rear = -1;
int main() {
  int n,data,i;
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    scanf("%d",&data);
    enqueue(data);
    display();
}</pre>
```