Implement Stack Using Array:

```
#include <iostream>
using namespace std;
int n=5;
int arr[5];
int top=-1;

void push(int value){
   if(top==n-1){
      cout<<"stack overflow"<<endl;
   }
   else{
      top++;
      arr[top]=value;
   }
}
int pop(){
   int valueToBePopped=-1;
   if(top==-1){
      cout<<"stack underflow"<<endl;
      return -1;
   }
   else{
      valueToBePopped=arr[top];
      top--;
   }
   return valueToBePopped;
}
int size(){
   return top+1;
}
bool isEmpty(){
   if(top==-1){
      return true;
   }
   else{
      return false;
   }
}
bool isFull(){
   if(top==n-1){
      return true;
   }
   else{
      return false;
   }
}
```

```
int topmostElement(){
   return arr[top];
}
int main() {
   push(30);
   cout<<pop()<<endl;
   cout<<pop()<<endl;
   push(40);
   push(50);
   cout<<size()<<endl;
   cout<<topmostElement()<<endl;
   cout<<isFull()<<endl;
   cout<<pop()<<endl;
   cout<<pop()<<endl;
   cout<<isEmpty();
   return 0;
}
```

Output:
30
stack underflow-1
2
50
0
50
40
1

Time:O(1)
Space:O(n)

Implement 2 Stacks in an Array:

```
#include <iostream>
using namespace std;
int n=5;
int arr[5];
int top1=-1;
int top2=n;

void push1(int value){
   if(top1+1==top2){
      cout<<"stack1 overflow";
   }
   else{
      top1++;
      arr[top1]=value;
   }
```

```
}
void push2(int value){
    if(top2-1==top1){
        cout<<"stack2 overflow";
    }
    else{
        top2--;
        arr[top2]=value;
    }
}
int pop1(){
    int valueToBePopped=-1;
    if(top1==-1){
        cout<<"stack1 underflow";
        return -1;
    }
    else{
        valueToBePopped=arr[top1];
        top1--;
    }
    return valueToBePopped;
}
int pop2(){
    int valueToBePopped=-1;
    if(top2==n){
        cout<<"stack2 underflow";
        return -1;
    }
    else{
        valueToBePopped=arr[top2];
        top2++;
    }
    return valueToBePopped;
}
int size1(){
    if(top1==-1){
        return 0;
    }
    else{
        return top1+1;
    }
}
int size2(){
    if(top2==n){
        return 0;
    }
    else{
        return n-top2;
```

```
    }
}
int topmostEle1(){
    if(top1==-1){
        return -1;
    }
    else{
        return arr[top1];
    }
}
int topmostEle2(){
    if(top2==n){
        return -1;
    }
    else{
        return arr[top2];
    }
}
bool isEmpty1(){
    if(top1==-1){
        return true;
    }
    else{
        return false;
    }
}
bool isEmpty2(){
    if(top2==n){
        return true;
    }
    else{
        return false;
    }
}
bool isFull1(){
    if(top1+1==top2){
        return true;
    }
    else{
        return false;
    }
}
bool isFull2(){
    if(top2-1==top1){
        return true;
    }
    else{
        return false;
```

```
   }
}
int main() {
   cout << pop2() << endl;
    push1(10);
    push1(20);
    cout << pop1() << endl;
    push2(30);
    push2(40);
    push2(50);
    push2(60);
    cout << pop1() << endl;
    push1(70);
    cout << isEmpty1() << endl;
    cout <<isFull2() << endl;
    cout << topmostEle1() << endl;
    cout << topmostEle2() << endl;
    cout << size2() << endl;

    return 0;
}
```

Output:
stack2 underflow-1
20
10
0
1
70
60
4

Time:O(1)
Space:O(n)

Implement Min Stack:

```
class MinStack {
public:
    stack<int>s,ms;
    MinStack() {

    }

    void push(int val) {
        s.push(val);
```

```
        if(ms.empty()||val<=ms.top()){
            ms.push(val);
        }
    }

    void pop() {
        if(s.top()==ms.top()){
            ms.pop();
        }
        s.pop();
    }

    int top() {
        return s.top();
    }

    int getMin() {
        return ms.top();
    }
};
```

Next Greater Element To The Right:

BruteForce:

```
 // vector<long long>ans;
       // for(int i=0;i<n;i++){
       //     int nge=-1;
       //     for(int j=i+1;j<n;j++){
       //         if(arr[j]>arr[i]){
       //             nge=arr[j];
       //             break;
       //         }
       //     }
       //     ans.push_back(nge);
       // }
       // return ans;
```

Time:O(n^2)(Time Limit Exceeded)
Space:O(1)

Optimal using stack:

```
 vector<long long>ans;
       stack<long long>s;
       for(int i=n-1;i>=0;i--){
          if(s.empty()){
              ans.push_back(-1);
```

```
            s.push(arr[i]);
        }
        else if(!s.empty()&&s.top()>arr[i]){
            ans.push_back(s.top());
            s.push(arr[i]);
        }
        else if(!s.empty()&&s.top()<=arr[i]){
            while(!s.empty()&&s.top()<=arr[i]){
                s.pop();
            }
            if(s.empty()){
            ans.push_back(-1);
            s.push(arr[i]);
        }
        else if(!s.empty()&&s.top()>arr[i]){
            ans.push_back(s.top());
            s.push(arr[i]);
        }
        }
    }
     reverse(ans.begin(),ans.end());
    return ans;
```

Time:O(n)
space:O(n)