**1 .Next Greater Element I**

*public static void* main(String[] args) {  
  
 *int*[] num1 = {4,1,2};  
 *int*[] num2 = {1,3,4,2};  
 *int*[] ans = *nextGreaterElement*(num1,num2);  
 System.***out***.println(Arrays.*toString*(ans));  
}  
  
*private static int*[] nextGreaterElement(*int*[] num1, *int*[] num2) {  
 Stack<Integer> s = *new* Stack<>();  
 HashMap<Integer,Integer> hmap = *new* HashMap<>();  
  
 *for* (*int* i=0;i< num2.length;i++){  
 *while*(!s.isEmpty()&& num2[i]>num2[s.peek()]){  
 hmap.put(num2[s.pop()], num2[i]);  
 }  
 s.push(i);  
 }  
  
 *while* (!s.isEmpty()){  
 hmap.put(num2[s.pop()],-1);  
 }  
  
 *for*(*int* i=0;i< num1.length;i++){  
 num1[i]= hmap.get(num1[i]);  
 }  
  
 *return* num1;  
}

2. Min Stack

*public class* stack {  
  
 Stack<Integer> s = *new* Stack<>();  
 Stack<Integer> ss = *new* Stack<>();  
  
 *void* push(*int* x){  
 s.push(x);  
 *if*(ss.size()==0||x<=ss.peek()){  
 ss.push(x);  
 }  
 }  
  
 *int* pop() {  
 *if*(s.isEmpty()){  
 *return* -1;  
 }  
 *int* ans = s.peek();  
 s.pop();  
 *if*(ans==ss.peek()){  
 ss.pop();  
 }  
 *return* ans ;  
 }  
  
 *int* getMin(){  
 *if*(s.isEmpty()){  
 *return* -1;  
 }  
 *return* ss.peek();  
 }  
  
 *int* top(){  
 *if*(s.isEmpty()){  
 *return* -1;  
 }  
 *return* s.peek();  
 }  
  
 @Override  
 *public* String toString() {  
 *return* "stack{" +  
 "s=" + s +  
 '}';  
 }  
  
 *public static void* main(String[] args) {  
 stack minStack = *new* stack();  
 minStack.push(-2) ;  
 minStack.push(0);  
 minStack.push(-3);  
 System.***out***.println(minStack);  
 System.***out***.println();  
 System.***out***.println(minStack.getMin());  
 minStack.pop();  
 System.***out***.println(minStack.top());  
 System.***out***.println(minStack.getMin());  
  
 }  
  
  
}

3.Implementing Stack using Queues

*public class* stack {  
  
 *Queue*<Integer> q1 = *new* LinkedList<>();  
 *Queue*<Integer> q2 = *new* LinkedList<>();  
  
 *public* stack(){  
  
 }  
  
 *public void* push(*int* val){  
 *if*(q1.isEmpty()){  
 q1.add(val);  
 *return*;  
 }  
 *while* (!q1.isEmpty()){  
 q2.add(q1.poll());  
 }  
 q1.add(val);  
 *while* (!q2.isEmpty()){  
 q1.add(q2.poll());  
 }  
 }  
  
 *public int* pop(){  
 *if*(q1.isEmpty()){  
 *return* -1;  
 }  
 *return* q1.poll();  
 }  
  
 *public int* top(){  
 *if*(q1.isEmpty()){  
 *return* -1;  
 }  
 *return* q1.peek();  
 }  
  
 *public boolean* empty(){  
 *if*(q1.isEmpty()){  
 *return true*;  
 }  
 *return false*;  
 }  
  
 @Override  
 *public* String toString() {  
 *return* "stack=" +  
 q1;  
 }  
  
 *public static void* main(String[] args) {  
 stack myStack = *new* stack();  
 myStack.push(1);  
 myStack.push(2);  
 System.***out***.println(myStack);  
 System.***out***.println(myStack.top());  
 System.***out***.println(myStack.pop());  
 System.***out***.println(myStack);  
 System.***out***.println(myStack.empty());  
 }  
}

4.Almost same as above

5.Maximum Nesting Depth Of Two Valid Parentheses Strings

// If you know answer you can add