# أسئلة على الستاك والكيو

## شعبة 3

محمد ابو صفط

1* ميثود داخل كلاس المين يفحص اذا كان السترنج المدخل من الاقواس صحيح ام لا :

*Input : s=”{{[][]()}}”

Output: true

*Input : s=”{}{”

Output: false

Solution:

```
public static boolean Check(String s){
    Stack<Character> stack = new Stack<>();
    for (int i = 0; i < s.length(); i++) {
        if(s.charAt(i)=='{'||s.charAt(i)=='('||s.charAt(i)=='[')
            stack.push(s.charAt(i));
        else{
            if(!stack.isEmpty()){

                if(s.charAt(i)=='}'&&!stack.peek().equals('{'))
                        return false;

                else if(s.charAt(i)==')'&&!stack.peek().equals('('))
                        return false;

                else if(s.charAt(i)==']'&&!stack.peek().equals('['))
                        return false;

                stack.pop();
            }
            else return false;

        }
    }
    return stack.isEmpty();
```

Write a method that will take two sorted stacks A and B (minimum value on top) and create one stack that is sorted (minimum value on top). You are allowed to use only the stack operations such as pop, push, size and peek. No other data structure such as arrays, lists, queues, ..etc are not allowed. You are allowed to use stacks only.
Note that elements on the stack can be compared using compareTo.

The method header:

public MyStack mergeStacks(MyStack A, MyStack B) {

```
public static Stack<Integer> merg(Stack<Integer> a , Stack <Integer> b){
    Stack<Integer> temp = new Stack<>();
    while(!a.isEmpty()&&!b.isEmpty()){
        if(a.peek()<b.peek())
            temp.push(a.pop());
        else temp.push(b.pop());
    }
    while(!a.isEmpty()){
            temp.push(a.pop());
    }
    while(!b.isEmpty()){
        temp.push(b.pop());
    }
    reverse(temp);
    return temp;
}
```

*3

Write a program to reverse a stack using recursion. You are not allowed to use loop constructs like while, for..etc, and you can only use the following ADT functions on Stack S:
isEmpty(S)
push(S)
pop(S)

```
public static void addBottom(Stack<Integer> s , int x){
    if(s.isEmpty())
        s.push(x);
    else {
        int a = s.pop();
        addBottom(s, x);
        s.push(a);
    }
}

public static void reverse(Stack<Integer> s){
    if(!s.isEmpty()) {
        int a = s.pop();
        reverse(s);
        addBottom(s, a);
    }
}
```

*4

Given an array, print the Next Greater Element (NGE) for every element. The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

*طبعا انه هاد الحل التايم اله O(n)

*Input : {0,2,3,1,4}

Output: 0 → 2 , 2 → 3 , 3 → 4 , 1 → 4 , 4 → -1

```
public static void PNG(int[] a , int n){
    Stack<Integer> stack = new Stack<>();
    int element;
    stack.push(a[0]);
    for (int i = 1; i < n; i++) {
        if(!stack.isEmpty()){
            element= stack.pop();
            while (element < a[i]){
                System.out.println(element+" ----> "+a[i]);
                if (stack.isEmpty())
                    break;
                element=stack.pop();
            }
            if (element > a[i])
                stack.push(element);
        }
        stack.push(a[i]);
    }
    while (!stack.isEmpty()){
        System.out.println(stack.pop()+" ----> "+-1);
    }
}
```

5* ميثود ياخد سترنج، لازم ما يكون عندي نفس الحرف كابيتال وسمول جنب بعض ويرجع السترنج الناتج :

*Input :”AasdD”

Output: ”s”

*Input :”AasdDSfs”

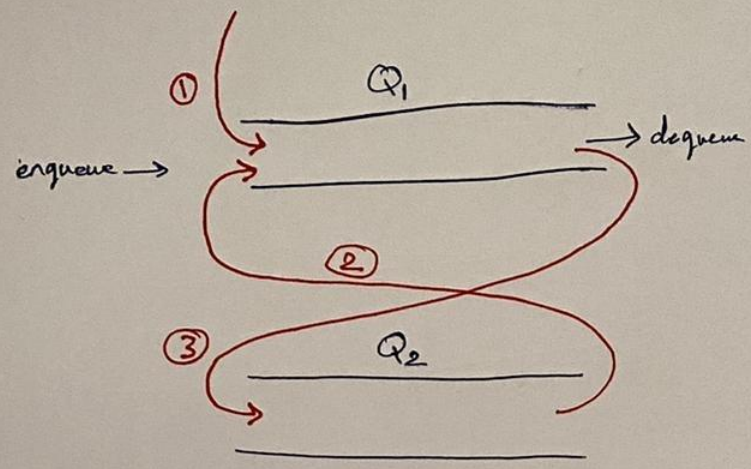
Output: ”fs”

```
public static String Aa(String s){
    Stack<Character> st = new Stack<>();
    for (int i = 0; i < s.length(); i++) {
        if(!st.isEmpty()&&Math.abs(s.charAt(i)-st.peek())==32)
            st.pop();
        else st.push(s.charAt(i));
    }
    String q = "";
    while(!st.isEmpty()){
        q= st.pop()+q;
    }
    return q;
}
```

implement a Stack class, using two Queues, you need to implement push(), pull() and peek() methods.

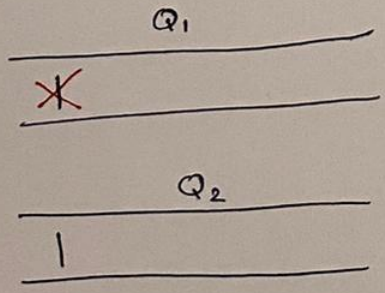
```
public class StackUsingQueue<E> {

    private MyQueue<E> Q1 = new MyQueue<E>();
    private MyQueue<E> Q2 = new MyQueue<E>();

    public void push(E e){
        Q1.enqueue(e);
        while(!Q2.isempty())
            Q1.enqueue(Q2.dequeue());
        while(!Q1.isempty())
            Q2.enqueue(Q1.dequeue());
    }

    public E pop(){
        if(Q2.isempty())
            throw new EmptyStackException();
        return Q2.dequeue();
    }

    public E peek(){
        if(Q2.isempty())
            throw new EmptyStackException();
        return Q2.peek();
    }
```

Push(x)

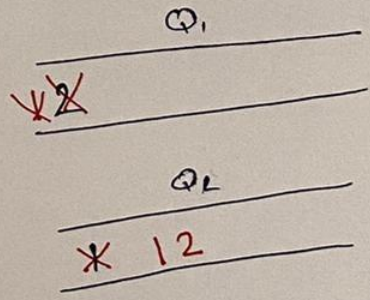



هاد اللي بكل الستاك

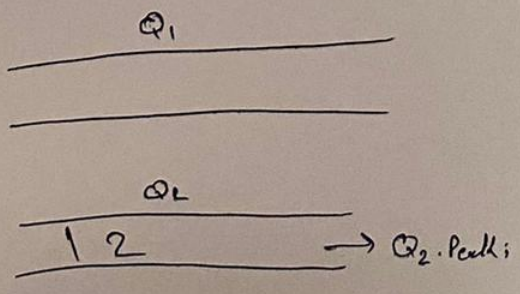
Push(1);



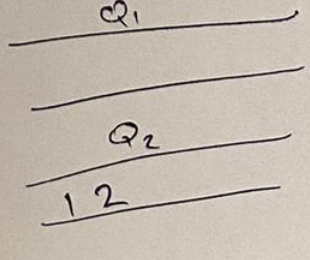

Push(2)




Peek();




Pop();




$Q_2$.dequeue();

يوجد حل اخر للسؤال انه يصير الشغل بميثود البوب

```java
public class MyStack<E> {

    private Queue<E> Q1 = new LinkedList<>();
    private Queue<E> Q2 = new LinkedList<>();

    public void push(E e){
        Q1.add(e);

    }

    public E pop(){
        while(Q1.size()!=1)
            Q2.add(Q1.remove());
        E temp = Q1.remove();
        while(!Q2.isEmpty())
            Q1.add(Q2.remove());
        return temp;
    }
```

*7

ميثود داخل كلاس ماي كيو يعمل ريفيرس للكيو باستخدام الركيرجن

```
public static void reverse(MyQueue Q){
    if(Q.isempty())
        return;
    else{
        Integer temp = (Integer)Q.dequeue();
        reverse(Q);
        Q.enqueue(temp);

    }

}
```