Problems:-

1.) Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

Note that an empty string is also considered valid.

****Example 1:****

****Input:**** "()"****Output:**** true

****Example 2:****

****Input:**** "()[]{}"****Output:**** true

****Example 3:****

****Input:**** "(]"****Output:**** false

****Example 4:****

****Input:**** "([)]"****Output:**** false

****Example 5:****

****Input:**** "{[]}"****Output:**** true

Solution :-

class Solution {

public boolean isValid(String s) {

if(s.length()%2!=0) return false;

Stack<Character> stack = new Stack<Character>();

Map<Character,Character> map = new HashMap<Character,Character>();

map.put(')', '(');

map.put('}', '{');

map.put(']', '[');

char charArray[]=s.toCharArray();

for(char c : charArray){

if(c=='(' || c=='{' || c=='[' ) stack.push(c);

if(c==')' || c=='}' || c==']'){

if(!stack.isEmpty()) {

if(stack.peek().equals(map.get(c))) stack.pop();

else return false;

}

}

}

if(stack.isEmpty()) return true;

else return false;

}

}

2.) Given a ****non-empty**** array of integers, every element appears twice except for one. Find that single one.  
  
****Note:**** Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?  
****Example 1:****

****Input:**** [2,2,1]****Output:**** 1

****Example 2:****

****Input:**** [4,1,2,1,2]****Output:**** 4