1. [Make the String Great](https://leetcode.com/problems/make-the-string-great/)

public class makeTheStringGreat{

    public static void main(String[] args) {

        String s = "leEeetcode";

        System.out.println(makeGood(s));

    }

    public static String makeGood(String s) {

        StringBuilder sb = new StringBuilder(s);

        int i = 0;

        while(i < sb.length() - 1){

            if(Math.abs(sb.charAt(i) - sb.charAt(i + 1)) == 32){

                sb.delete(i, i + 2);

                i = 0;

            }else{

                i++;

            }

        }

        return sb.toString();

    }

}

1. [Remove all adjacent duplicates in a String](https://leetcode.com/problems/remove-all-adjacent-duplicates-in-string/)

public class removeAllDuplicateFromString {

    public static void main(String[] args) {

        String s = "abbaca";

        System.out.println(removeDuplicates(s));

    }

    public static String removeDuplicates(String s) {

        int i = 0;

        char[] res = s.toCharArray();

        for (int j = 0; j < s.length(); ++j, ++i) {

            res[i] = res[j];

            if (i > 0 && res[i - 1] == res[i])

                i -= 2;

        }

        return new String(res, 0, i);

    }

}

1. [Online Stock Span](https://leetcode.com/problems/online-stock-span/)

import java.util.Stack;

public class onlineStockSpan {

    public static void main(String[] args) {

        StockSpanner obj = new StockSpanner();

        System.out.println(obj.next(100));

        System.out.println(obj.next(80));

        System.out.println(obj.next(60));

        System.out.println(obj.next(70));

        System.out.println(obj.next(60));

        System.out.println(obj.next(75));

        System.out.println(obj.next(85));

    }

    static class StockSpanner {

        Stack<int[]> stack;

        public StockSpanner() {

            stack = new Stack<>();

        }

        public int next(int price) {

            int res = 1;

            while(!stack.isEmpty() && stack.peek()[0] <= price){

                res += stack.pop()[1];

            }

            stack.push(new int[]{price, res});

            return res;

        }

    }

}

1. [Time Needed to buy tickets](https://leetcode.com/problems/time-needed-to-buy-tickets/)

public class timeNeedeToBuyStock {

    public static void main(String[] args) {

        int[] tickets = {2,3,2};

        int k = 2;

        System.out.println(timeRequiredToBuy(tickets, k));

    }

        public static int timeRequiredToBuy(int[] tickets, int k) {

          int i=0;int j=0;

            int second=0;

            while(tickets[k]>0){

                j=0;

                while(j<tickets.length) {

                    if(tickets[j]>0&&tickets[k]!=0) {

                        tickets[j] = tickets[j] - 1;

                        second++;

                    }

                    j++;

                }

            }

           // System.out.println(second);

            return second;

        }

}

1. [Product of the last K Numbers](https://leetcode.com/problems/product-of-the-last-k-numbers/)

import java.util.ArrayList;

import java.util.List;

public class productOfLastKNumber {

    public static void main(String[] args) {

        ProductOfNumbers obj = new ProductOfNumbers();

        obj.add(3);

        obj.add(0);

        obj.add(2);

        obj.add(5);

        obj.add(4);

        System.out.println(obj.getProduct(2));

        System.out.println(obj.getProduct(3));

        System.out.println(obj.getProduct(4));

        obj.add(8);

        System.out.println(obj.getProduct(2));

    }

    static class ProductOfNumbers {

        List<Integer> list;

        public ProductOfNumbers() {

            list = new ArrayList<>();

        }

        public void add(int num) {

            if(num == 0){

                list = new ArrayList<>();

            }else{

                if(list.size() == 0){

                    list.add(num);

                }else{

                    list.add(list.get(list.size() - 1) \* num);

                }

            }

        }

        public int getProduct(int k) {

            if(k > list.size()){

                return 0;

            }else if(k == list.size()){

                return list.get(list.size() - 1);

            }else{

                return list.get(list.size() - 1) / list.get(list.size() - k - 1);

            }

        }

    }

}

1. [Largest Rectangle in Histogram](https://leetcode.com/problems/largest-rectangle-in-histogram/description/)
2. [Sliding Window Maximum](https://leetcode.com/problems/sliding-window-maximum/)
3. Design a circular queue data structure with the following operations: **enQueue**, **deQueue**, **Front**, **Rear**, and **isEmpty**.