DS-PROBLEM CHALLENGES

Submitted by-

Gauraang Goel 21CSU319

Kartik Singh 21CSU233

Sanskar Tamta 21CSU232

Dev Singh 21CSU514

ARRAYS

Q-11 **Problem**

James visits a restaurant, looks at the menu, and realizes that there is no price on it. Since he wanted to know the prices before he orders, he looked up the restaurant online and found n different versions of the menu. He knew from experience that usually the menu which has the maximum number of items that have the maximum price on that item between the menus is the most updated one and if there are multiple menus with that condition the one with the maximum average price is the most updated one. Help him find the most updated menu.

In other words, a price on an item is good if it is the maximum price on that item among all menus, and a menu is the most updated one if it has the maximum number of good prices on it.  
If there are multiple menus with the maximum number of good prices, the menu with the higher price average is the most updated one.

**Input format**

* The first line contains integers n and m that denote the number of menus and the number of items on each menu respectively.
* The next n line each contains m integers represented as Aij, the jth price on the ith menu.

**Output format**

Print a single number denoting the number of the most updated menu.

It is guaranteed that the answer is unique.

CODE

import java.util.\*;

public class Main

{

    static Scanner sc =new Scanner(System.in);

    static void input(int a[][], int n, int m)

    {

        System.out.println("Please enter the values to be added");

        for(int i=0 ; i<n ; i++)

        {

            for(int j=0 ; j<m ; j++)

            {

                a[i][j] = sc.nextInt();

            }

            System.out.println();

        }

    }

    static void calculateMaxValue(int a[][],int result[], int n, int m)

    {

        int i, j, t, max, k;

        for( j=0 ; j<m ; j++)

        {

            k = 0;

            t = 0;

            max = a[0][j];

            for( i=1 ; i<n ; i++)

            {

                if( max<a[i][j] )

                {

                    max = a[i][j];

                    k = i;

                }

                else if( max == a[i][j] )

                {

                    t = i;

                }

            }

            if( t>0 )

            {

                result[k]++;

                result[t]++;

            }

            else

            {

                result[k]++;

            }

        }

    }

    static void average(int a[][],int result[],int p[], int n, int m, int t)

    {

        int i, j, z, w, min, s = 0;

        int sum[];

        sum = new int[1000];

        for( w=0 ; w<t ; w++)

        {

            z = p[w];

            sum[z] = 0;

            for( j=0 ; j<m ; j++)

            {

                sum[w] = sum[w] + a[z][j];

            }

            sum[w] = sum[w]/m;

        }

        min = sum[0];

        for( i=0 ; i<t ; i++)

        {

            if( min>sum[i] )

            {

                min = sum[i];

                s = i+1;

            }

        }

        System.out.println(s);

    }

    static void caluclateReturn(int a[][], int result[], int n, int m)

    {

        int i, k, max, p[], t = 0;

        p = new int[1000];

        k = 0;

        max = result[0];

        for(i=1;i<n;i++)

        {

            if(max<result[i])

            {

                max = result[i];

                k = i+1;

            }

            else if(max == result[i])

            {

                p[i-1] = i;

                t++;

            }

        }

        if(t == 0)

        {

            System.out.println("Correct Menu " + k);

        }

        else

        {

            average(a, result, p, n, m, t);

        }

    }

    public static void main(String[] args)

    {

        int a[][] = new int[1000][1000];

        int result[] = new int[1000];

        int n, m, i;

        System.out.println("Enter the number of menus");

        n=sc.nextInt();

        System.out.println("Enter the number of items");

        m=sc.nextInt();

        input(a,n,m);

        for(i=0;i<1000;i++)

        {

            result[i] = 0;

        }

        calculateMaxValue(a,result,n,m);

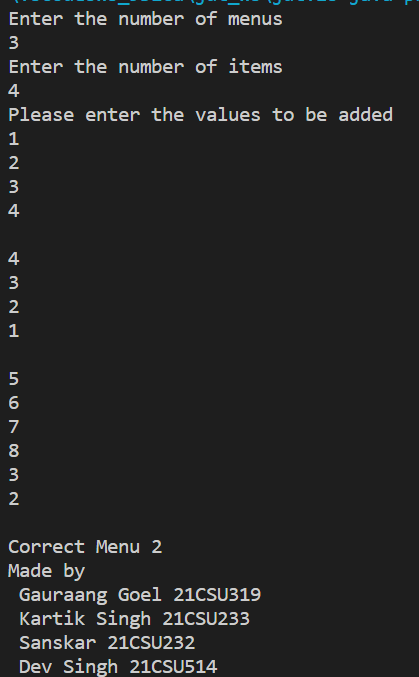
        caluclateReturn(a,result,n,m);

        System.out.println("Made by \n Gauraang Goel 21CSU319 \n Kartik Singh 21CSU233 \n Sanskar 21CSU232 \n Dev Singh 21CSU514");

    }

}

Output-



LINKED LIST

A linked list contains N nodes numbered from 1 to N. The tail of the list points to head of the list i.e. the linked list is circular in nature.

For when N=5

1->2->3->4,

^-------5<---'

An integer K is also given to you.You start counting from head and when you reach Kth node in the circular list you remove that node. For eg. if K=7 we start from head i.e. 1 right now nd move K steps ahead till we reach node numbered 3 in circular list 1->2->3->4->5->1->2->[3] <---- we remove this node

now the new linked list looks like this

1->2->4,

^---5<---'

Node numbered 3 has been removed and now we start counting from the next node. This process is repeated until only one Node is left.

Your task is to determine the number of the last node.

**INPUT FORMAT:**

Line 1: Number of test cases T Line 2: 2 Integers **N K** , N=Number of nodes in circular list and K=Number of steps to count

**CONSTRAINTS:** 1<=T<=10 1<=N<=1000 1<=K<=10^8

**OUTPUT FORMAT:**

T lines,containing Number of the last node remaining for each case

**NOTE: This question must be solved using linked list.**

CODE

import java.util.\*;

class second

{

static class Node

{

    int data;

    Node next;

    Node(int x)

    {

        data = x;

        next = null;

    }

};

static void printList(Node head)

{

    if (head == null)

        return;

    Node temp = head;

    do

    {

        System.out.print( temp.data + "->");

        temp = temp.next;

    }

    while (temp != head);

    System.out.println(head.data );

}

static Node deleteK(Node head\_ref, int k)

{

    Node head = head\_ref;

    if (head == null)

        return null;

    Node curr = head, prev=null;

    while (true)

    {

        if (curr.next == head && curr == head)

            break;

        printList(head);

        for (int i = 0; i < k; i++)

        {

            prev = curr;

            curr = curr.next;

        }

        if (curr == head)

        {

            prev = head;

            while (prev.next != head)

                prev = prev.next;

            head = curr.next;

            prev.next = head;

            head\_ref = head;

        }

        else if (curr.next == head)

        {

            prev.next = head;

        }

        else

        {

            prev.next = curr.next;

        }

    }

    return head;

}

static Node insertNode(Node head\_ref, int x)

{

    Node head = head\_ref;

    Node temp = new Node(x);

    if (head == null)

    {

        temp.next = temp;

        head\_ref = temp;

        return head\_ref;

    }

    else

    {

        Node temp1 = head;

        while (temp1.next != head)

            temp1 = temp1.next;

        temp1.next = temp;

        temp.next = head;

    }

    return head;

}

public static void main(String args[])

{

    Node head = null;

    Scanner sc =new Scanner(System.in);

    int n, val;

    System.out.println("Enter No of nodes to be inserted");

    n=sc.nextInt();

    int i=1;

    while(i<=n)

    {  System.out.println("Enter value for "+ i + " node");

       val=sc.nextInt();

    head = insertNode(head, val);

    i=i+1;

    }

    System.out.println("Enter number of k th node to be deleted ");

    int k = sc.nextInt();

    head = deleteK(head, k);

    System.out.println("Made by \n Gauraang Goel 21CSU319 \n Kartik Singh 21CSU233 \n Sanskar 21CSU232 \n Dev Singh 21CSU514");

}

}

**OUTPUT**

