Write a program in Java to right rotate an array by 5 steps ?

class RotateArray {

public void rotate(int[] nums, int k) {

if(k > nums.length)

k=k%nums.length;

int[] result = new int[nums.length];

for(int i=0; i < k; i++){

result[i] = nums[nums.length-k+i];

}

int j=0;

for(int i=k; i<nums.length; i++){

result[i] = nums[j];

j++;

}

System.arraycopy( result, 0, nums, 0, nums.length );

}

}

public class Main

{

public static void main(String[] args) {

RotateArray r = new RotateArray();

int arr[] = { 1, 2, 3, 4, 5, 6, 7 };

r.rotate(arr, 5);

for(int i=0;i<arr.length;i++){

System.out.print(arr[i]+" ");

}

}

}

Write a program in Java to find the fourth smallest element in an unsorted list ?

class KthSmallst

{

int kthSmallest(int arr[], int l, int r, int k)

{

if (k > 0 && k <= r - l + 1)

{

int pos = randomPartition(arr, l, r);

if (pos-l == k-1)

return arr[pos];

if (pos-l > k-1)

return kthSmallest(arr, l, pos-1, k);

return kthSmallest(arr, pos+1, r, k-pos+l-1);

}

return Integer.MAX\_VALUE;

}

void swap(int arr[], int i, int j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

int partition(int arr[], int l, int r)

{

int x = arr[r], i = l;

for (int j = l; j <= r - 1; j++)

{

if (arr[j] <= x)

{

swap(arr, i, j);

i++;

}

}

swap(arr, i, r);

return i;

}

int randomPartition(int arr[], int l, int r)

{

int n = r-l+1;

int pivot = (int)(Math.random()) \* (n-1);

swap(arr, l + pivot, r);

return partition(arr, l, r);

}

}

public class Main

{

public static void main(String[] args) {

KthSmallst ob = new KthSmallst();

int arr[] = {12, 3, 5, 7, 4, 19, 26};

int n = arr.length,k = 4;

System.out.println("K'th smallest element is "+ ob.kthSmallest(arr, 0, n-1, k));

}

}

Write a program in Java to find the sum of n number of elements in the range of L and R where 0 <= L <= R <= n-1 ?

import java.util.Scanner;

class demo{

public static void main(String args[]) throws Exception{

Scanner sc = new Scanner(System.in);

int range\_n = sc.nextInt();

//int l = sc.nextInt();

//int r = sc.nextInt();

int sum=0;

for(int i=0;i<range\_n;i++) {

sum = sum + i;

}

System.out.println(sum);

}

}

Write a program in Java to multiply two matrices ?

import java.util.Scanner;

class demo{

public static void main(String args[]) throws Exception{

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int m = sc.nextInt();

int[][] arr = new int[n][m];

int[][] arr1 = new int[n][m];

int[][] arr3 = new int[n][m];

for(int i=0;i<n;i++) {

for(int j=0;j<m;j++) {

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

//System.out.println(arr[i][j]);

}

}

for(int i=0;i<n;i++) {

for(int j=0;j<m;j++) {

arr1[i][j] = sc.nextInt();

//System.out.println(arr[i][j]);

}

}

for(int i=0;i<n;i++) {

for(int j=0;j<m;j++) {

arr3[i][j] = arr[i][j]\*arr1[i][j];

//System.out.println(arr[i][j]);

}

}

for(int i=0;i<n;i++) {

for(int j=0;j<m;j++) {

System.out.println(arr3[i][j]);

}

}

}

}

Write a program in Java to delete the first occurrence of a key in a singly linked list ?

import java.io.\*;

import java.util.\*;

public class LinkedList

{

Node head;

        static class Node{

                int data;

                Node next;

                Node(int d){

                        data = d;

                        next = null;

                }

        }

        public static LinkedList insert(LinkedList list, int data){

                Node new\_node = new Node(data);

                new\_node.next = null;

                if (list.head == null){

                        list.head = new\_node;

                }

                else{

                        Node last = list.head;

                        while (last.next != null){

                            last = last.next;

                        }

                        last.next = new\_node;

                }

                return list;

        }

    public static void printList(LinkedList list){

                Node currNode = list.head;

                System.out.print("LinkedList: ");

                while (currNode != null){

                        System.out.print(currNode.data + " ");

                        currNode = currNode.next;

                }

                System.out.println();

        }

        public static LinkedList deleteByKey(LinkedList list, int key){

                Node currNode = list.head, prev = null;

                if(currNode != null && currNode.data == key){

                        list.head = currNode.next;

                        System.out.println(key + " found and deleted");

                        return list;

                }

                while (currNode != null && currNode.data != key){

                        prev = currNode;

                        currNode = currNode.next;

                }

                if (currNode != null){

                        prev.next = currNode.next;

                        System.out.println(key + " found and deleted");

                }

                if (currNode == null){

                        System.out.println(key + " not found");

                }

                return list;

        }

        public static void main(String[] args)

        {

                LinkedList list = new LinkedList();

                list = insert(list, 1);

                list = insert(list, 2);

                list = insert(list, 3);

                list = insert(list, 4);

                list = insert(list, 5);

                list = insert(list, 6);

                list = insert(list, 7);

                list = insert(list, 8);

                printList(list);

                deleteByKey(list, 1);

                printList(list);

        }

}

Write a program in Java to insert a new element in a sorted circular linked list?

public class DLL{

     Node head;

class Node{

         int data;

         Node prev;

         Node next;

Node(int d){

data = d;

}

     }

public void push(int new\_data){

Node new\_Node = new Node(new\_data);

new\_Node.next = head;

         new\_Node.prev = null;

if (head != null)

             head.prev = new\_Node;

head = new\_Node;

     }

public void InsertAfter(Node prev\_Node, int new\_data){

if (prev\_Node == null){

             System.out.println("The given previous node cannot be NULL ");

             return;

         }

Node new\_node = new Node(new\_data);

new\_node.next = prev\_Node.next;

prev\_Node.next = new\_node;

new\_node.prev = prev\_Node;

if (new\_node.next != null)

             new\_node.next.prev = new\_node;

     }

     void append(int new\_data){

Node new\_node = new Node(new\_data);

   Node last = head;

new\_node.next = null;

if (head == null)

{

             new\_node.prev = null;

             head = new\_node;

             return;

         }

while (last.next != null)

             last = last.next;

last.next = new\_node;

new\_node.prev = last;

     }

public void printlist(Node node){

         Node last = null;

         System.out.println("Traversal in forward Direction");

         while (node != null){

             System.out.print(node.data + " ");

             last = node;

             node = node.next;

         }

         System.out.println();

         System.out.println("Traversal in reverse direction");

         while (last != null){

             System.out.print(last.data + " ");

             last = last.prev;

         }

     }

public static void main(String[] args){

DLL dll = new DLL

dll.append(6);

dll.push(7);

dll.push(1);

dll.append(4);

dll.InsertAfter(dll.head.next, 8);

   System.out.println("Created DLL is: ");

         dll.printlist(dll.head);

     }

}

Write a program in Java to traverse a doubly linked list in the forward and backward directions ?

public class DLL

{

     Node head;

class Node

{

         int data;

         Node prev;

         Node next;

Node(int d)

{

data = d;

}

     }

public void push(int new\_data)

     {

Node new\_Node = new Node(new\_data);

new\_Node.next = head;

         new\_Node.prev = null;

if (head != null)

             head.prev = new\_Node;

head = new\_Node;

     }

public void InsertAfter(Node prev\_Node, int new\_data)

     {

if (prev\_Node == null)

{

             System.out.println("The given previous node cannot be NULL ");

             return;

         }

Node new\_node = new Node(new\_data);

new\_node.next = prev\_Node.next;

prev\_Node.next = new\_node;

new\_node.prev = prev\_Node;

if (new\_node.next != null)

             new\_node.next.prev = new\_node;

     }

     void append(int new\_data)

     {

Node new\_node = new Node(new\_data);

   Node last = head;

new\_node.next = null;

if (head == null)

{

             new\_node.prev = null;

             head = new\_node;

             return;

         }

while (last.next != null)

             last = last.next;

last.next = new\_node;

new\_node.prev = last;

     }

public void printlist(Node node)

     {

         Node last = null;

         System.out.println("Traversal in forward Direction");

         while (node != null)

{

             System.out.print(node.data + " ");

             last = node;

             node = node.next;

         }

         System.out.println();

         System.out.println("Traversal in reverse direction");

         while (last != null)

{

             System.out.print(last.data + " ");

             last = last.prev;

         }

     }

public static void main(String[] args)

     {

DLL dll = new DLL

dll.append(6);

dll.push(7);

dll.push(1);

dll.append(4);

dll.InsertAfter(dll.head.next, 8);

   System.out.println("Created DLL is: ");

         dll.printlist(dll.head);

     }

}

Write a program in Java to insert and remove elements in a stack ?

import java.util.Stack;

class demo{

public static void main(String args[]) throws Exception{

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

Stack<Integer> s = new Stack<Integer>();

s.add(1);

s.add(2);

s.add(3);

s.add(4);

s.add(5);

System.out.println("elemnt inserted in stack");

System.out.println(s);

s.remove();

System.out.println(s);

}

}

Write a program in Java to insert and remove elements in a queue ?

import java.util.LinkedList;

import java.util.Queue;

class demo{

public static void main(String args[]) throws Exception{

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

Queue<Integer> s = new LinkedList<Integer>();

s.add(1);

s.add(2);

s.add(3);

s.add(4);

s.add(5);

System.out.println("elemnt inserted in stack");

System.out.println(s);

s.remove(3);

System.out.println(s);

}

}