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

Source code :

package smpili;

public class arrayof5 {

public static void rotateArrayByKSteps(int[] nums, int k) {

if (nums == null || nums.length <= 1 || k % nums.length == 0) {

return; // No rotation needed

}

int n = nums.length;

k = k % n;

*reverseArray*(nums, 0, n - 1);

*reverseArray*(nums, 0, k - 1);

*reverseArray*(nums, k, n - 1);

}

public static void reverseArray(int[] nums, int start, int end) {

while (start < end) {

int temp = nums[start];

nums[start] = nums[end];

nums[end] = temp;

start++;

end--;

}

}

public static void main(String[] args) {

int[] nums = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int k = 5;

System.***out***.println("Original Array:");

*printArray*(nums);

*rotateArrayByKSteps*(nums, k);

System.***out***.println("Array after right rotation by " + k + " steps:");

*printArray*(nums);

}

public static void printArray(int[] nums) {

for (int num : nums) {

System.***out***.print(num + " ");

}

System.***out***.println();

}

}

}

Output:

A screen shot of a computer

Description automatically generated with medium confidence

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

Source code:

package smpili;

import java.util.Arrays;

public class fouthsmall2 {

public static int findFourthSmallest(int[] nums) {

if (nums == null || nums.length < 4) {

throw new IllegalArgumentException("Invalid input");

}

Arrays.*sort*(nums);

return nums[3];

}

public static void main(String[] args) {

int[] nums = {9, 5, 7, 1, 3, 6, 2, 8, 4};

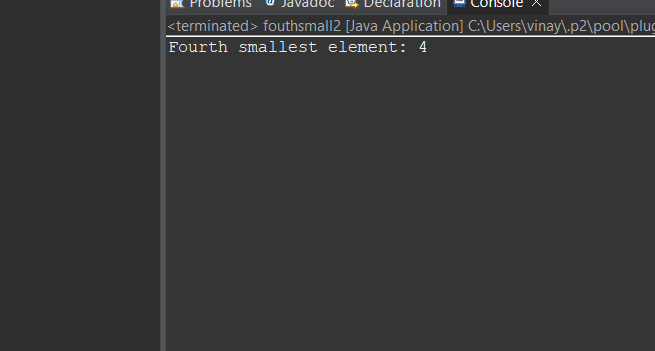
int fourthSmallest = *findFourthSmallest*(nums);

System.***out***.println("Fourth smallest element: " + fourthSmallest);

}

}

OUTPUT:

**

3.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 ?

package smpili;

public class leftright {

//public class SumInRange {

public static int sumInRange(int[] nums, int L, int R) {

if (nums == null || L < 0 || R >= nums.length || L > R) {

throw new IllegalArgumentException("Invalid input");

}

int sum = 0;

for (int i = L; i <= R; i++) {

sum += nums[i];

}

return sum;

}

public static void main(String[] args) {

int[] nums = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};

int L = 2;

int R = 6;

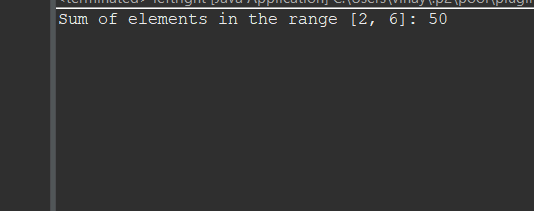
int sum = *sumInRange*(nums, L, R);

System.***out***.println("Sum of elements in the range [" + L + ", " + R + "]: " + sum);

}

}

Output:



*4.* Write a program in Java to multiply two matrices?

*Source code:*

package smpili;

public class multiply2matrics {

//public class MatrixMultiplication {

public static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2) {

int rows1 = matrix1.length;

int cols1 = matrix1[0].length;

int cols2 = matrix2[0].length;

if (cols1 != matrix2.length) {

throw new IllegalArgumentException("Invalid matrix dimensions");

}

int[][] result = new int[rows1][cols2];

for (int i = 0; i < rows1; i++) {

for (int j = 0; j < cols2; j++) {

for (int k = 0; k < cols1; k++) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

return result;

}

public static void printMatrix(int[][] matrix) {

for (int[] row : matrix) {

for (int num : row) {

System.***out***.print(num + " ");

}

System.***out***.println();

}

}

public static void main(String[] args) {

int[][] matrix1 = {{1, 2, 3}, {4, 5, 6}};

int[][] matrix2 = {{7, 8}, {9, 10}, {11, 12}};

int[][] result = *multiplyMatrices*(matrix1, matrix2);

System.***out***.println("Matrix 1:");

*printMatrix*(matrix1);

System.***out***.println("Matrix 2:");

*printMatrix*(matrix2);

System.***out***.println("Result of matrix multiplication:");

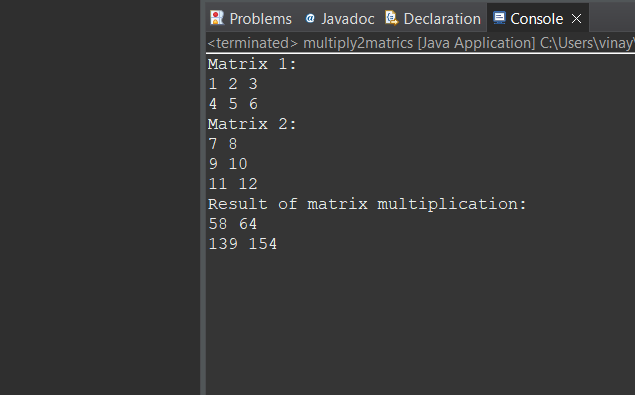
*printMatrix*(result);

}

}

}

*OUTPUT:*

**

5Write a program in Java to delete the first occurrence of a key in a singly linked list?

Source Code:

package smpili;

class Node {

int data;

Node next;

Node(int data) {

this.data = data;

next = null;

}

}

public class LinkedList {

private Node head;

public void insert(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

}

}

public void deleteFirstOccurrence(int key) {

if (head == null) {

return;

}

if (head.data == key) {

head = head.next;

return;

}

Node prev = head;

Node current = head.next;

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

prev = current;

current = current.next;

}

if (current != null) {

prev.next = current.next;

}

}

public void display() {

Node temp = head;

while (temp != null) {

System.***out***.print(temp.data + " ");

temp = temp.next;

}

System.***out***.println();

}

public static void main(String[] args) {

LinkedList linkedList = new LinkedList();

linkedList.insert(10);

linkedList.insert(20);

linkedList.insert(30);

linkedList.insert(40);

linkedList.insert(50);

System.***out***.println("Original linked list:");

linkedList.display();

int key = 30;

linkedList.deleteFirstOccurrence(key);

System.***out***.println("Linked list after deleting first occurrence of " + key + ":");

linkedList.display();

}

}

OUTPUT:

*A screenshot of a computer

Description automatically generated with medium confidence*

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

Source Code:

class Node {

int data;

Node next;

Node(int data) {

this.data = data;

next = null;

}

}

public class SortedCircularLinkedList {

private Node head;

public void insert(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

head.next = head; // Make it circular

return;

}

Node current = head;

Node prev = null;

do {

prev = current;

current = current.next;

if (data <= current.data || current == head) {

break;

}

} while (current != head);

prev.next = newNode;

newNode.next = current;

if (data < head.data) {

head = newNode;

}

}

public void display() {

if (head == null) {

return;

}

Node current = head;

do {

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

current = current.next;

} while (current != head);

System.out.println();

}

public static void main(String[] args) {

SortedCircularLinkedList circularList = new SortedCircularLinkedList();

circularList.insert(10);

circularList.insert(20);

circularList.insert(30);

circularList.insert(40);

circularList.insert(50);

System.out.println("Original circular linked list:");

circularList.display();

int newElement = 25;

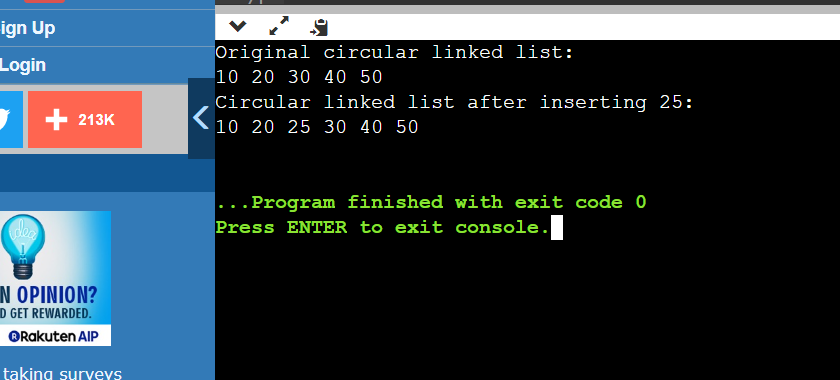
circularList.insert(newElement);

System.out.println("Circular linked list after inserting " + newElement + ":");

circularList.display();

}

}Ouput;

**

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

Source code:

class Node {

int data;

Node prev;

Node next;

Node(int data) {

this.data = data;

prev = null;

next = null;

}

}

public class DoublyLinkedList {

private Node head;

private Node tail;

public void insert(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

tail = newNode;

} else {

tail.next = newNode;

newNode.prev = tail;

tail = newNode;

}

}

public void displayForward() {

Node current = head;

System.out.println("Doubly linked list (Forward):");

while (current != null) {

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

current = current.next;

}

System.out.println();

}

public void displayBackward() {

Node current = tail;

System.out.println("Doubly linked list (Backward):");

while (current != null) {

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

current = current.prev;

}

System.out.println();

}

public static void main(String[] args) {

DoublyLinkedList doublyLinkedList = new DoublyLinkedList();

doublyLinkedList.insert(10);

doublyLinkedList.insert(20);

doublyLinkedList.insert(30);

doublyLinkedList.insert(40);

doublyLinkedList.insert(50);

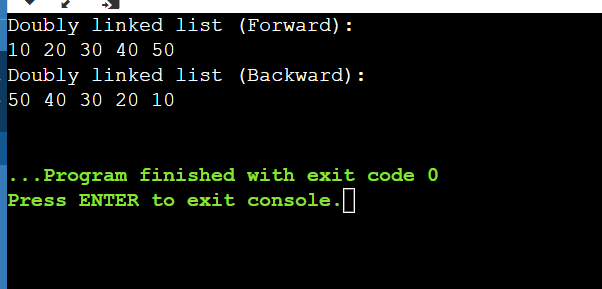
doublyLinkedList.displayForward();

doublyLinkedList.displayBackward();

}

}

*Output:*

**

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

*Source code:*

package smpili;

import java.util.Stack;

public class doublely {

//public class StackExample {

public static void main(String[] args) {

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

// Inserting elements into the stack

stack.push(10);

stack.push(20);

stack.push(30);

stack.push(40);

stack.push(50);

System.***out***.println("Stack elements: " + stack);

// Removing elements from the stackS

int removedElement = stack.pop();

System.***out***.println("Removed element: " + removedElement);

// Peek the top element without removing it

int topElement = stack.peek();

System.***out***.println("Top element: " + topElement);

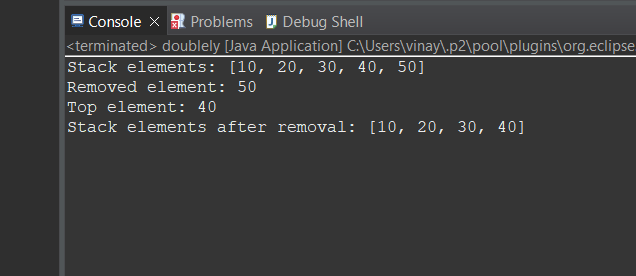
System.***out***.println("Stack elements after removal: " + stack);

}

} }

}

*Outout:*



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

*Source code:*

import java.util.LinkedList;

import java.util.Queue;

public class doublely {

//public class QueueExample {

public static void main(String[] args) {

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

// Inserting elements into the queue

queue.offer(10);

queue.offer(20);

queue.offer(30);

queue.offer(40);

queue.offer(50);

System.***out***.println("Queue elements: " + queue);

// Removing elements from the queue

int removedElement = queue.poll();

System.***out***.println("Removed element: " + removedElement);

// Peek the front element without removing it

int frontElement = queue.peek();

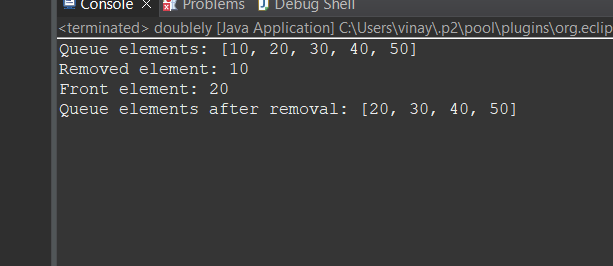
System.***out***.println("Front element: " + frontElement);

System.***out***.println("Queue elements after removal: " + queue);

}

}

*Output:*

**