FACULTY OF ENGINEERING AND TECHNOLOGY

BACHELOR OF TECHNOLOGY

COMPETITIVE CODING
(303105259)

SEMESTER IV
Computer Science & Engineering Department

Laboratory Manual

CERTIFICATE

This is to certify that

Mr./Miss **……………………………………………………………..…**

*with Enrollment No.***…………………..…………………………** has

successfully completed his/her laboratory experiments **COMPETITIVE CODING (303105259)** *from the department of* **Computer Science and Engineering** *during the academic year* ***2023-2024.***

योगः कर्मसु कौशलम्

PARUL UNIVERSITY

Date of Submission ….…………….

Staff In charge …..……………

Head of Department……………..……

TABLE OF CONTENTS

10	Write a Program to Understand and implement Tree traversals i.e. Pre-Order Post-Order, In-Order						
11	Write a Program to check whether two trees are mirrored or not.						
12	Write a Program to find the height of a binary tree.						
13	Write a program for Lowest Common Ancestors.						
14	Write a Program to Build BST						
15	Write a Program for Building a Function ISVALID to VALIDATE BST						
16	Write a Program to Traverse a Tree using Level Order Traversal						
17	Write a Program to perform Boundary Traversal on BST						
18	Write a Program to view a tree from left View						
19	Write a Program for a basic hash function in a programming language of your choice. Demonstrate its usage to store and retrieve key-value pairs.						
20	Write a Program to Implement Two sums using HASHMAP						
21	Write a Program to find Distinct substrings in a string						

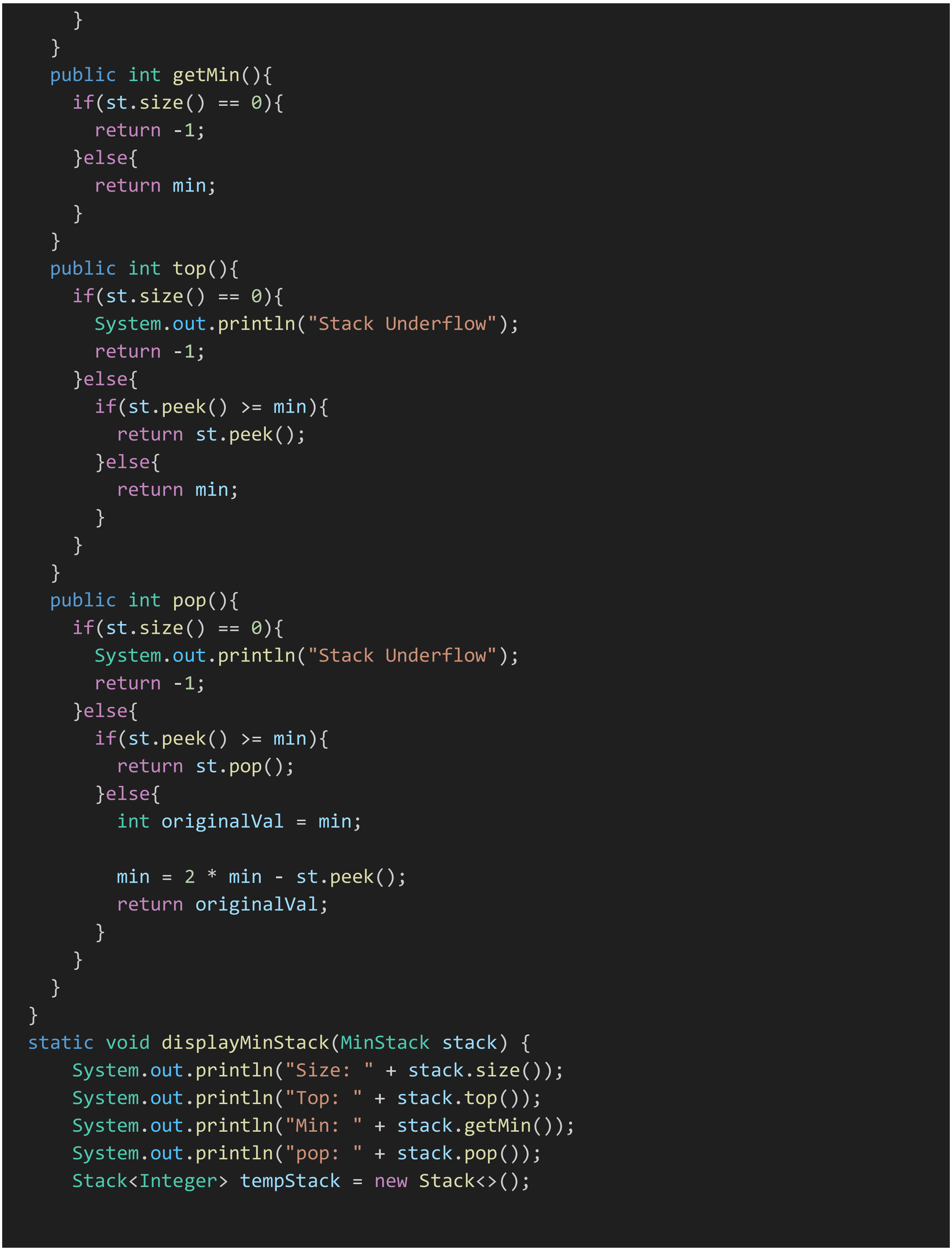
PRACTICAL NO: 1

AIM: Write a program for implementing a MINSTACK which should support operations like push, pop, overflow, underflow, display

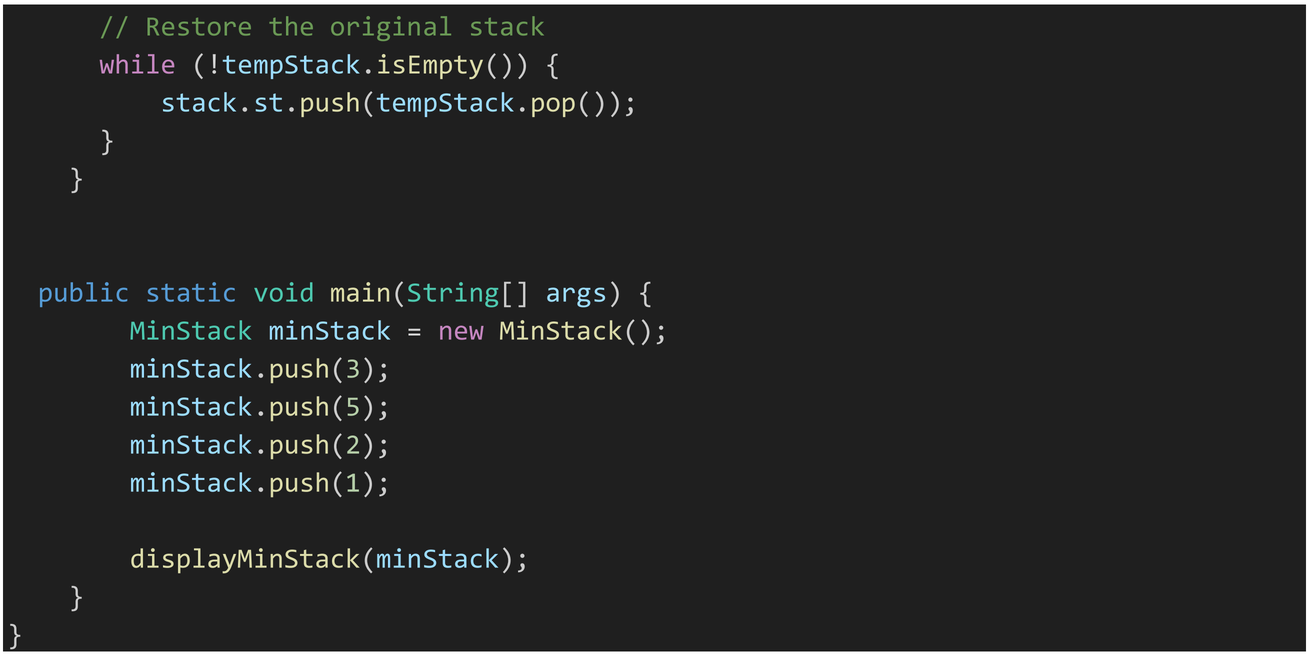
- **Construct a stack of N-capacity**
- **Push elements**
- **Pop elements**
- **Top element**
- **Retrieve the min element from the stack**

SOURCE CODE:

```
import java.util.*;

public class Main {
  public static class MinStack{
    Stack<Integer> st;
    int min;

    public MinStack(){
      st = new Stack<>();
    }
    public boolean isEmpty(){
      return st.size() == 0;
    }
    public int size(){
      return st.size();
    }

    public void push(int val){
      if(st.size() == 0){
        st.push(val);
        min = val;
      }else if(val > st.peek() ) {
        st.push(val);
      }else{
        st.push(val + val - min);
        min = val;
```

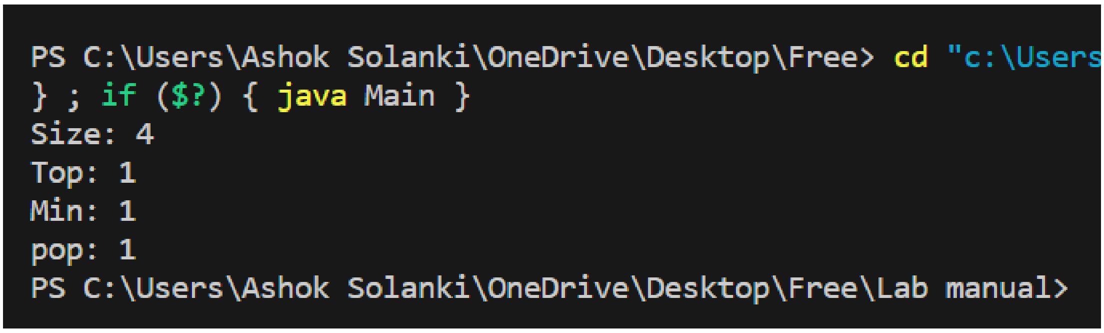


```
    }
  }
  public int getMin(){
    if(st.size() == 0){
      return -1;
    }else{
      return min;
    }
  }
  public int top(){
    if(st.size() == 0){
      System.out.println("Stack Underflow");
      return -1;
    }else{
      if(st.peek() >= min){
        return st.peek();
      }else{
        return min;
      }
    }
  }
  public int pop(){
    if(st.size() == 0){
      System.out.println("Stack Underflow");
      return -1;
    }else{
      if(st.peek() >= min){
        return st.pop();
      }else{
        int originalVal = min;

        min = 2 * min - st.peek();
        return originalVal;
      }
    }
  }
}
static void displayMinStack(MinStack stack) {
    System.out.println("Size: " + stack.size());
    System.out.println("Top: " + stack.top());
    System.out.println("Min: " + stack.getMin());
    System.out.println("pop: " + stack.pop());
    Stack<Integer> tempStack = new Stack<>();
```



```
      // Restore the original stack
      while (!tempStack.isEmpty()) {
          stack.st.push(tempStack.pop());
      }
    }


  public static void main(String[] args) {
        MinStack minStack = new MinStack();
        minStack.push(3);
        minStack.push(5);
        minStack.push(2);
        minStack.push(1);

        displayMinStack(minStack);
    }
}
```

OUTPUT:



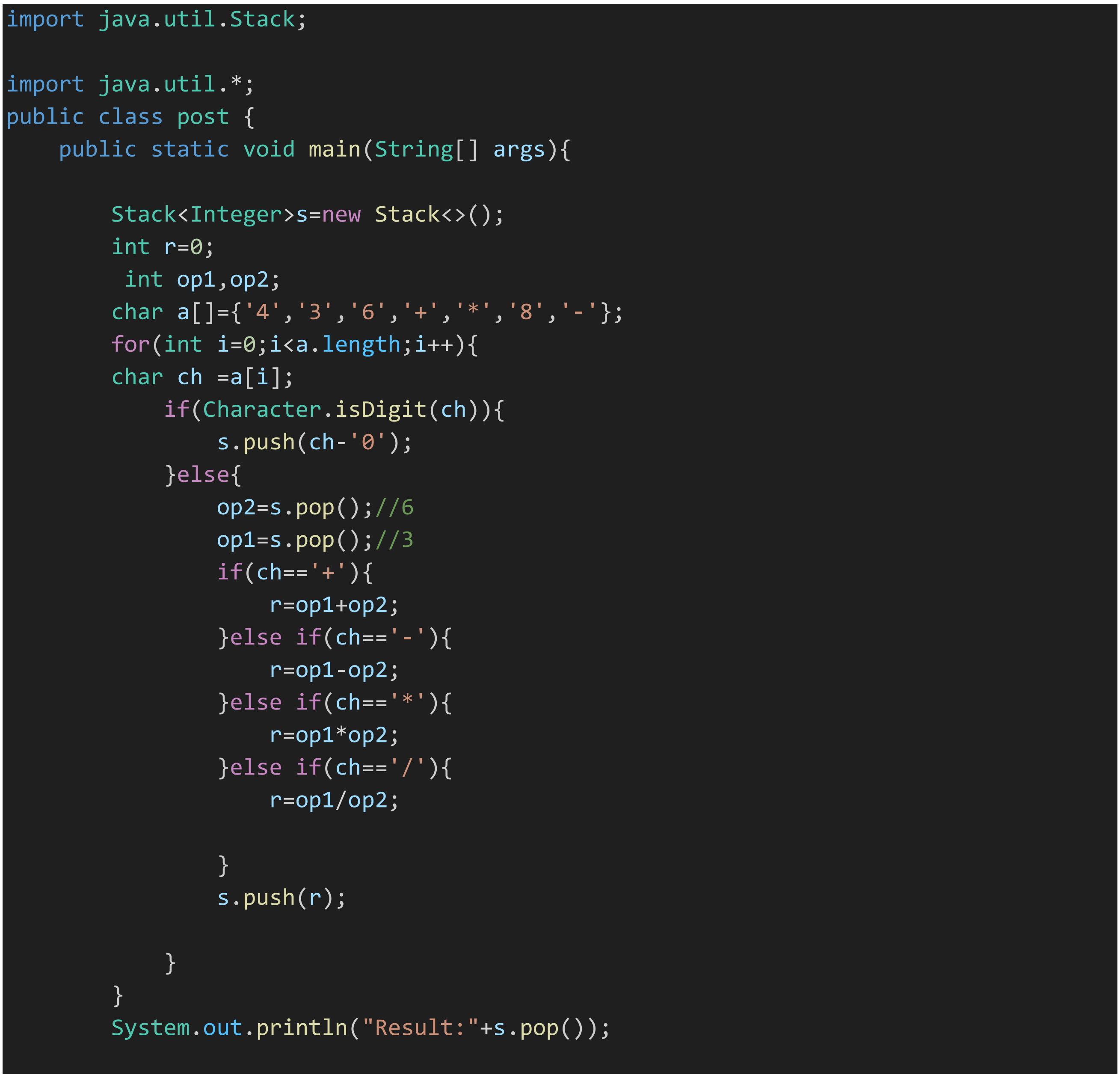
```
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free> cd "c:\Users
} ; if ($?) { java Main }
Size: 4
Top: 1
Min: 1
pop: 1
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free\Lab manual>
```

CONCLUSION: The above code is successfully executed in Lab.

PRACTICAL NO: 2

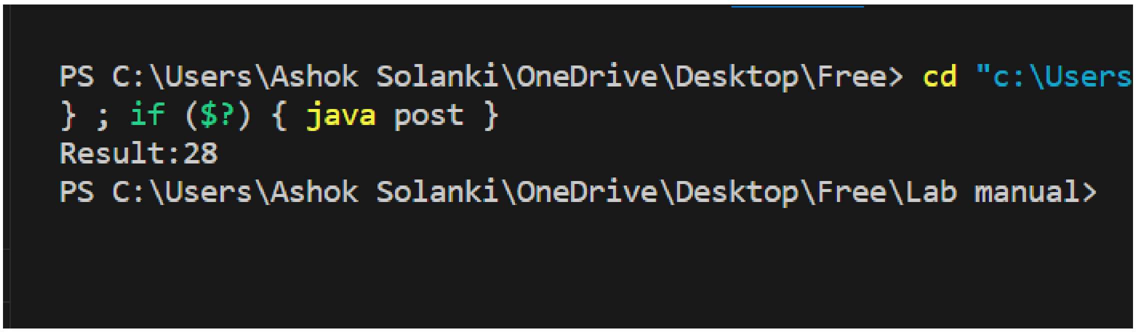
AIM: Write a program to deal with real-world situations where Stack data structure is widely used Evaluation of expression: Stacks are used to evaluate expressions, especially in languages that use postfix or prefix notation. Operators and operands are pushed onto the stack, and operations are performed based on the LIFO principle**.**

SOURCE CODE:



```java
import java.util.Stack;

import java.util.*;
public class post {
    public static void main(String[] args){

        Stack<Integer>s=new Stack<>();
        int r=0;
         int op1,op2;
        char a[]={'4','3','6','+','*','8','-'};
        for(int i=0;i<a.length;i++){
        char ch =a[i];
            if(Character.isDigit(ch)){
                s.push(ch-'0');
            }else{
                op2=s.pop();//6
                op1=s.pop();//3
                if(ch=='+'){
                    r=op1+op2;
                }else if(ch=='-'){
                    r=op1-op2;
                }else if(ch=='*'){
                    r=op1*op2;
                }else if(ch=='/'){
                    r=op1/op2;

                }
                s.push(r);

            }
        }
        System.out.println("Result:"+s.pop());
    }
}
```

OUTPUT:



```
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free> cd "c:\Users
} ; if ($?) { java post }
Result:28
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free\Lab manual>
```

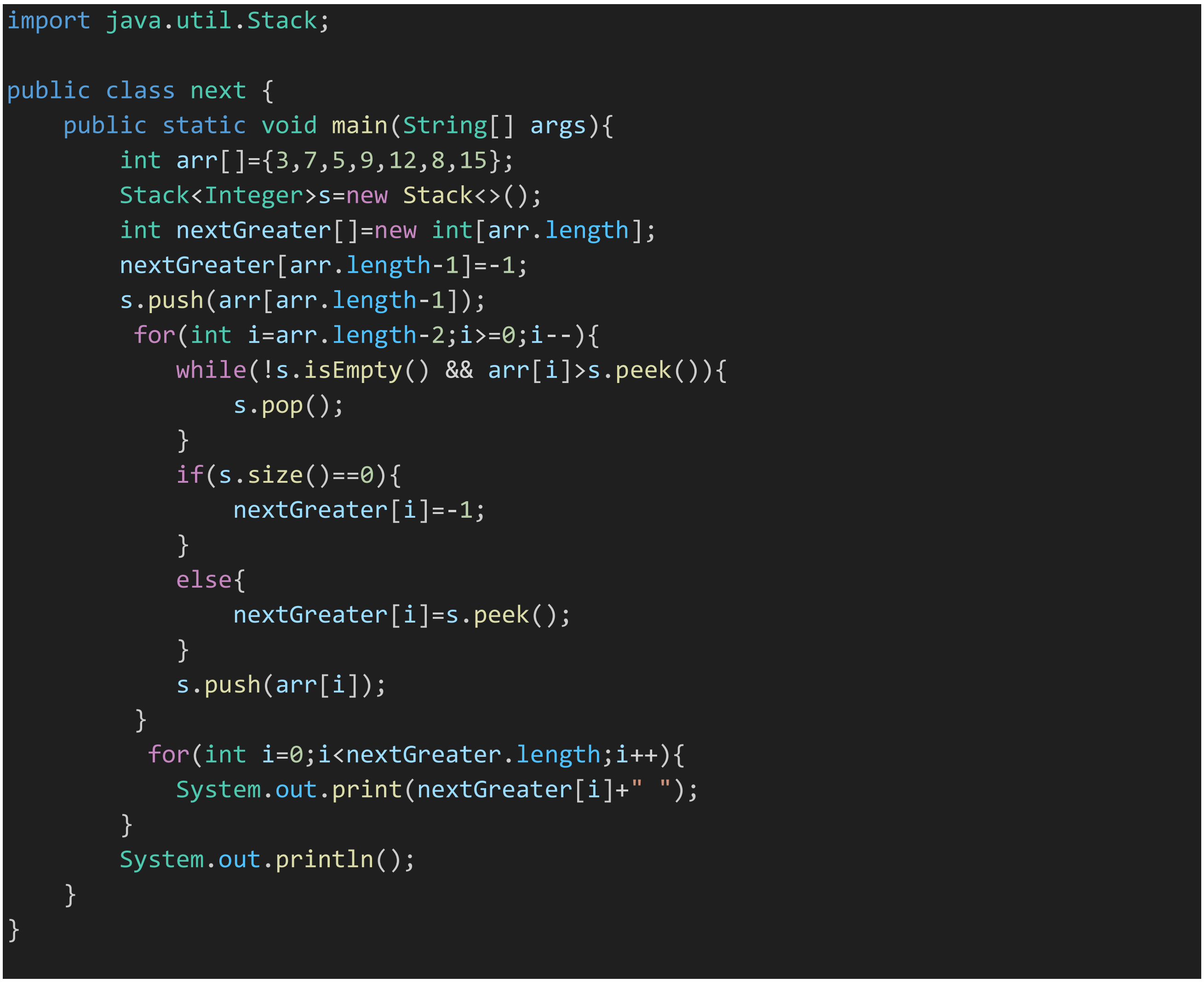
CONCLUSION: The above code is successfully executed in Lab.

PRACTICAL NO: 3

AIM: Write a program for finding NGE NEXT GREATER ELEMENT from an array **.**

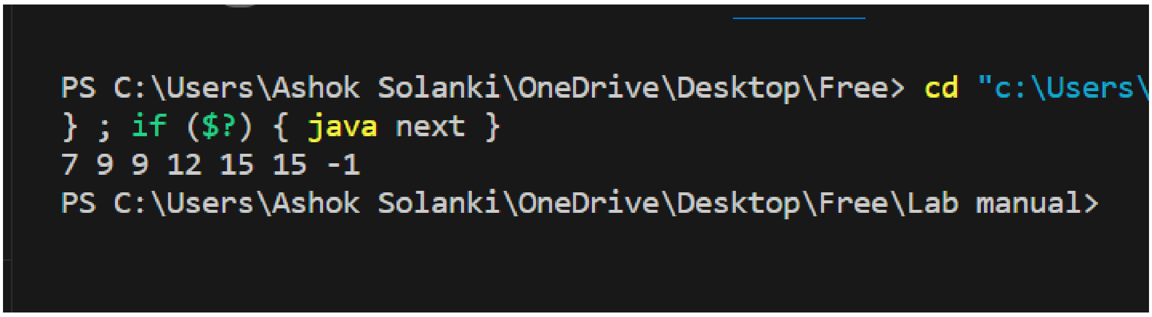
PROBLEM STATEMENT: You are given an array of integers. For each element in the array, find the next greater element to its right. If there is no greater element, consider -1 as the next greater element.

SOURCE CODE:



```
import java.util.Stack;

public class next {
    public static void main(String[] args){
        int arr[]={3,7,5,9,12,8,15};
        Stack<Integer>s=new Stack<>();
        int nextGreater[]=new int[arr.length];
        nextGreater[arr.length-1]=-1;
        s.push(arr[arr.length-1]);
         for(int i=arr.length-2;i>=0;i--){
            while(!s.isEmpty() && arr[i]>s.peek()){
                s.pop();
            }
            if(s.size()==0){
                nextGreater[i]=-1;
            }
            else{
                nextGreater[i]=s.peek();
            }
            s.push(arr[i]);
         }
          for(int i=0;i<nextGreater.length;i++){
            System.out.print(nextGreater[i]+" ");
        }
        System.out.println();
    }
}
```

OUTPUT:



```
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free> cd "c:\Users\
} ; if ($?) { java next }
7 9 9 12 15 15 -1
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free\Lab manual>
```

CONCLUSION: The above code is successfully executed in Lab.

PRACTICAL NO: 4

AIM: Write a program to design a circular queue(k) which Should implement the below functions

a. Enqueue

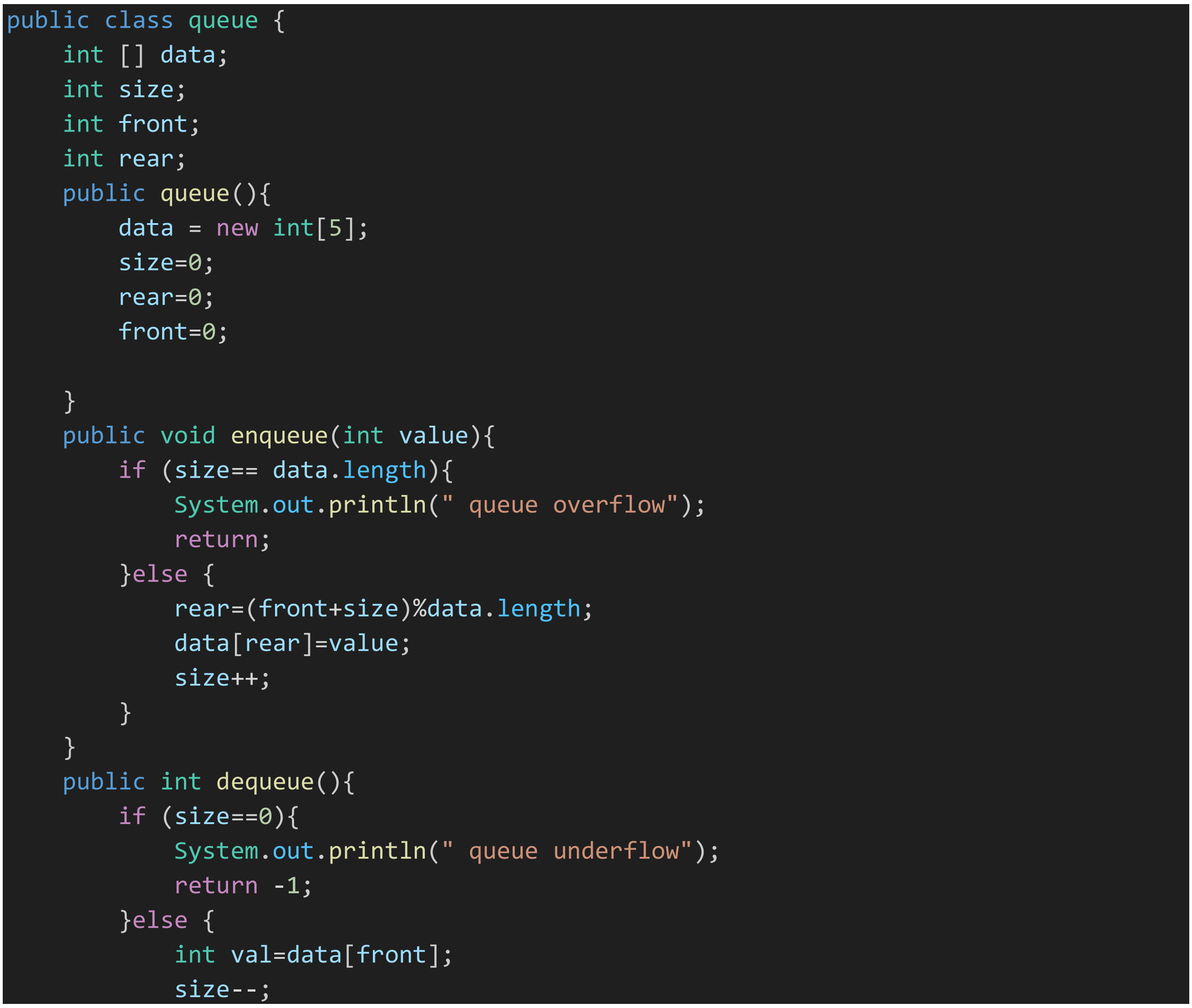
b. Dequeue

c. Front

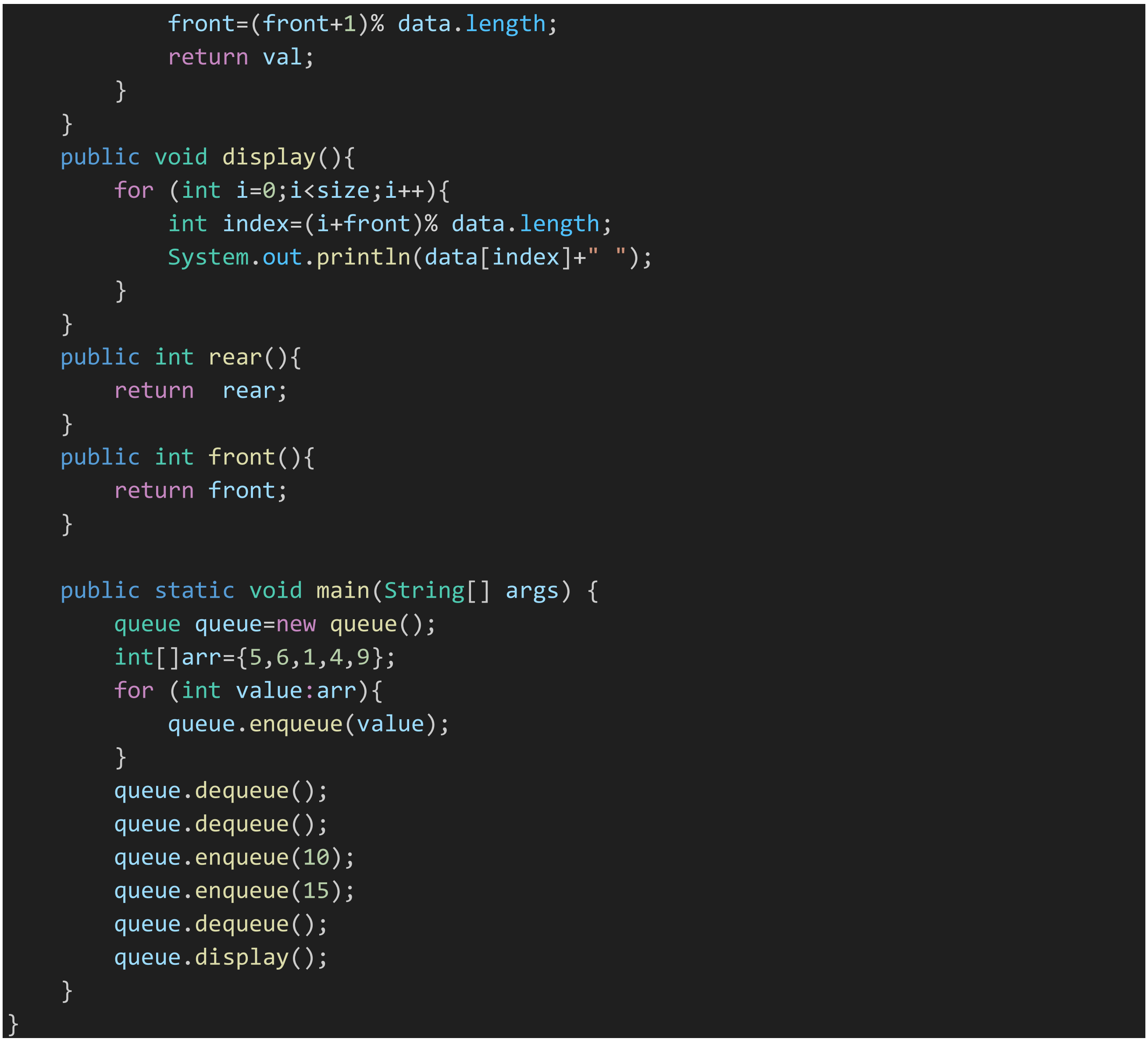
d. Rear

PROBLEM STATEMENT: You are tasked with designing a circular queue with the following functionalities: Enqueue, Dequeue, Front, and Rear.

SOURCE CODE:

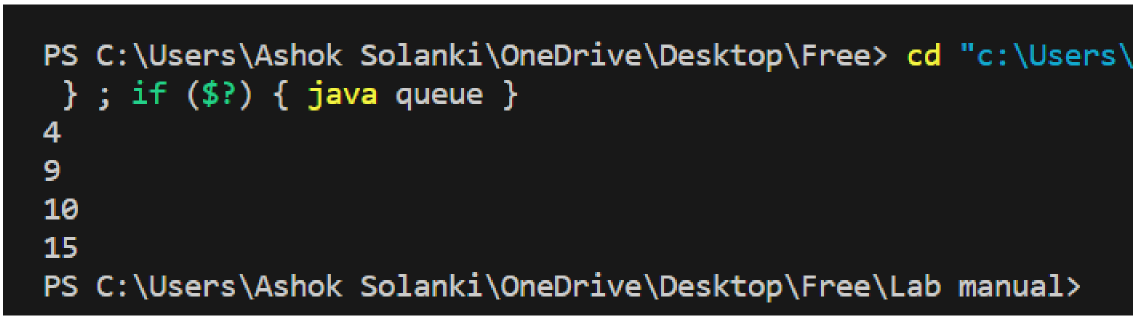


```java
public class queue {
    int [] data;
    int size;
    int front;
    int rear;
    public queue(){
        data = new int[5];
        size=0;
        rear=0;
        front=0;

    }
    public void enqueue(int value){
        if (size== data.length){
            System.out.println(" queue overflow");
            return;
        }else {
            rear=(front+size)%data.length;
            data[rear]=value;
            size++;
        }
    }
    public int dequeue(){
        if (size==0){
            System.out.println(" queue underflow");
            return -1;
        }else {
            int val=data[front];
            size--;
```



```
            front=(front+1)% data.length;
            return val;
        }
    }
    public void display(){
        for (int i=0;i<size;i++){
            int index=(i+front)% data.length;
            System.out.println(data[index]+" ");
        }
    }
    public int rear(){
        return  rear;
    }
    public int front(){
        return front;
    }

    public static void main(String[] args) {
        queue queue=new queue();
        int[]arr={5,6,1,4,9};
        for (int value:arr){
            queue.enqueue(value);
        }
        queue.dequeue();
        queue.dequeue();
        queue.enqueue(10);
        queue.enqueue(15);
        queue.dequeue();
        queue.display();
    }
}
```

OUTPUT:



```
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free> cd "c:\Users\
 } ; if ($?) { java queue }
4
9
10
15
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free\Lab manual>
```

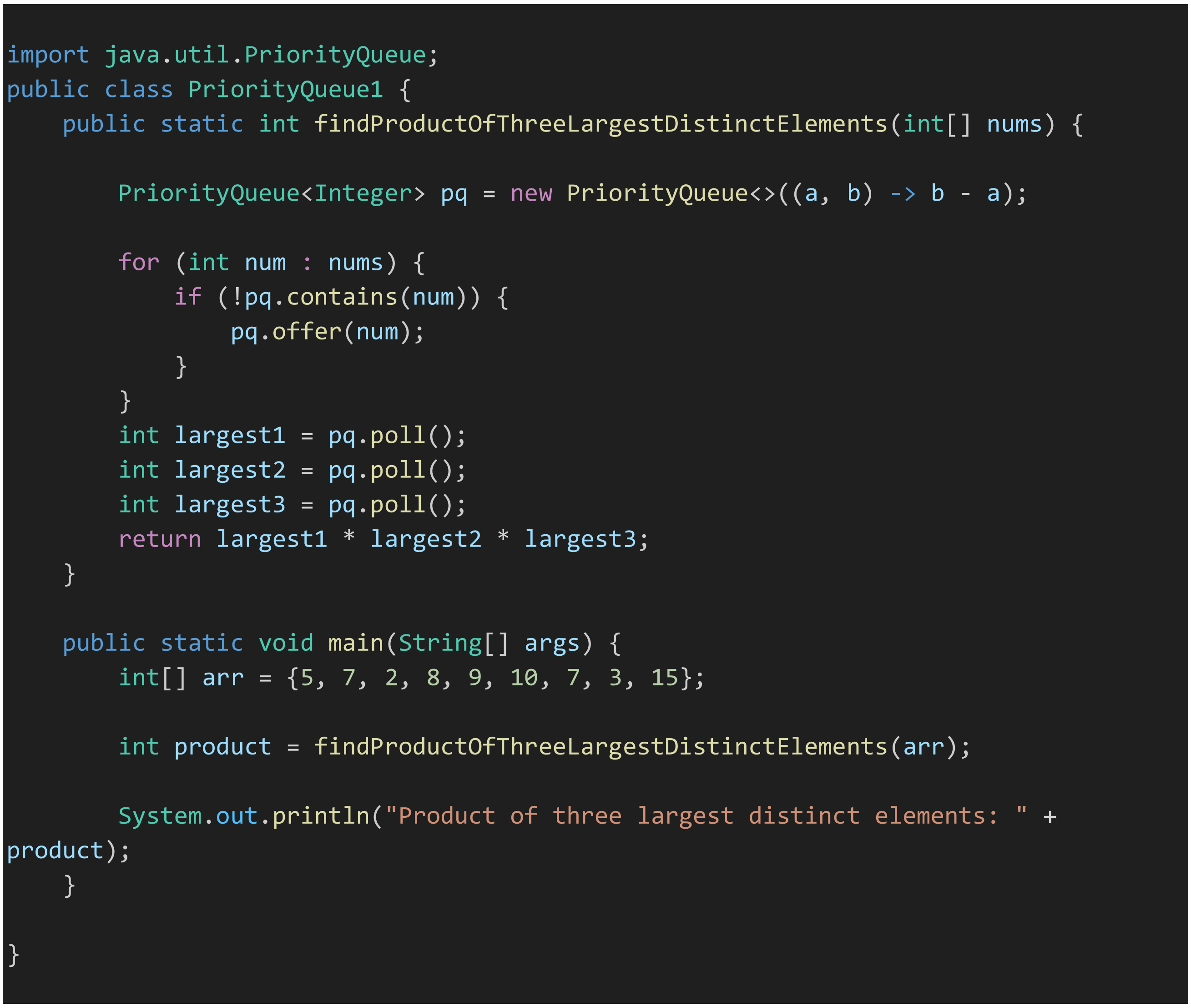
CONCLUSION: The above code is successfully executed in Lab.

PRACTICAL NO: 5

AIM: Write a Program for finding the Product of the three largest Distinct Elements. Use a Priority Queue to efficiently find and remove the largest elements .

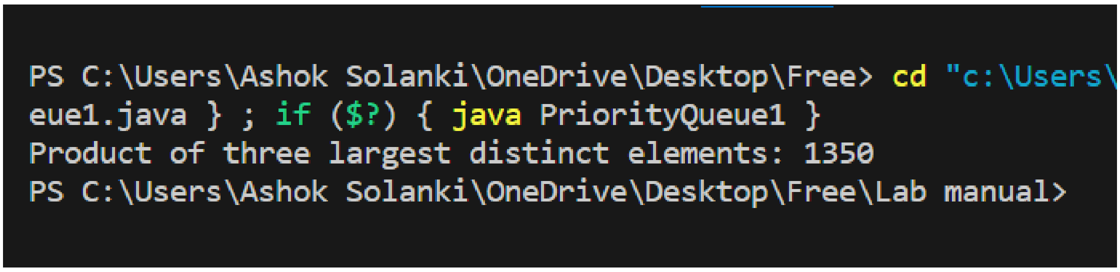
PROBLEM STATEMENT: You are given an array of integers. Write a Java program to find the product of the three largest distinct elements in the array. Implement this program using a Priority Queue to efficiently find and remove the largest elements.

SOURCE CODE:



```java
import java.util.PriorityQueue;
public class PriorityQueue1 {
    public static int findProductOfThreeLargestDistinctElements(int[] nums) {

        PriorityQueue<Integer> pq = new PriorityQueue<>((a, b) -> b - a);

        for (int num : nums) {
            if (!pq.contains(num)) {
                pq.offer(num);
            }
        }
        int largest1 = pq.poll();
        int largest2 = pq.poll();
        int largest3 = pq.poll();
        return largest1 * largest2 * largest3;
    }

    public static void main(String[] args) {
        int[] arr = {5, 7, 2, 8, 9, 10, 7, 3, 15};

        int product = findProductOfThreeLargestDistinctElements(arr);

        System.out.println("Product of three largest distinct elements: " +
product);
    }

}
```

OUTPUT:



```
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free> cd "c:\Users\
eue1.java } ; if ($?) { java PriorityQueue1 }
Product of three largest distinct elements: 1350
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free\Lab manual>
```

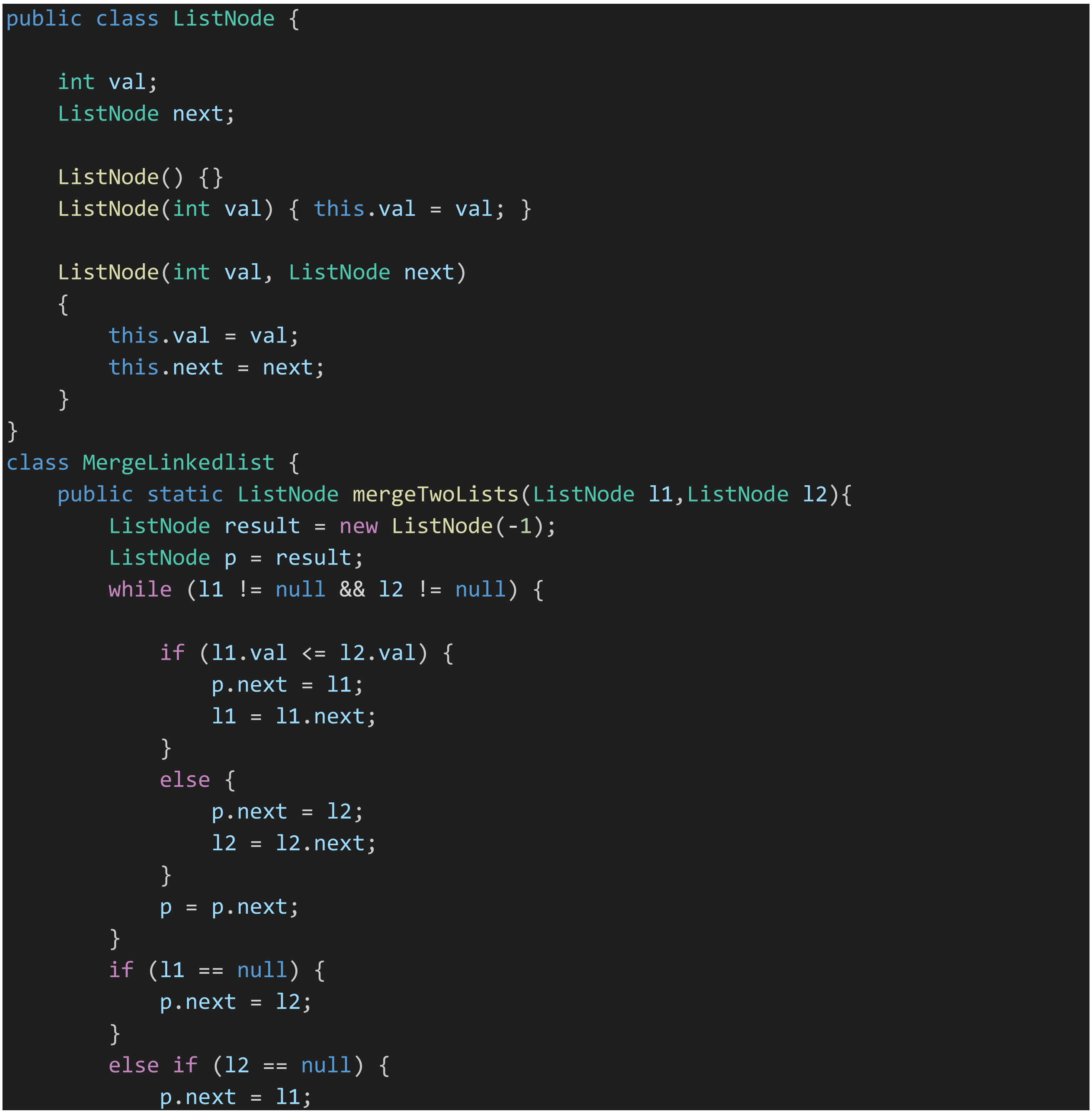
CONCLUSION: The above code is successfully executed in Lab.

PRACTICAL NO: 6

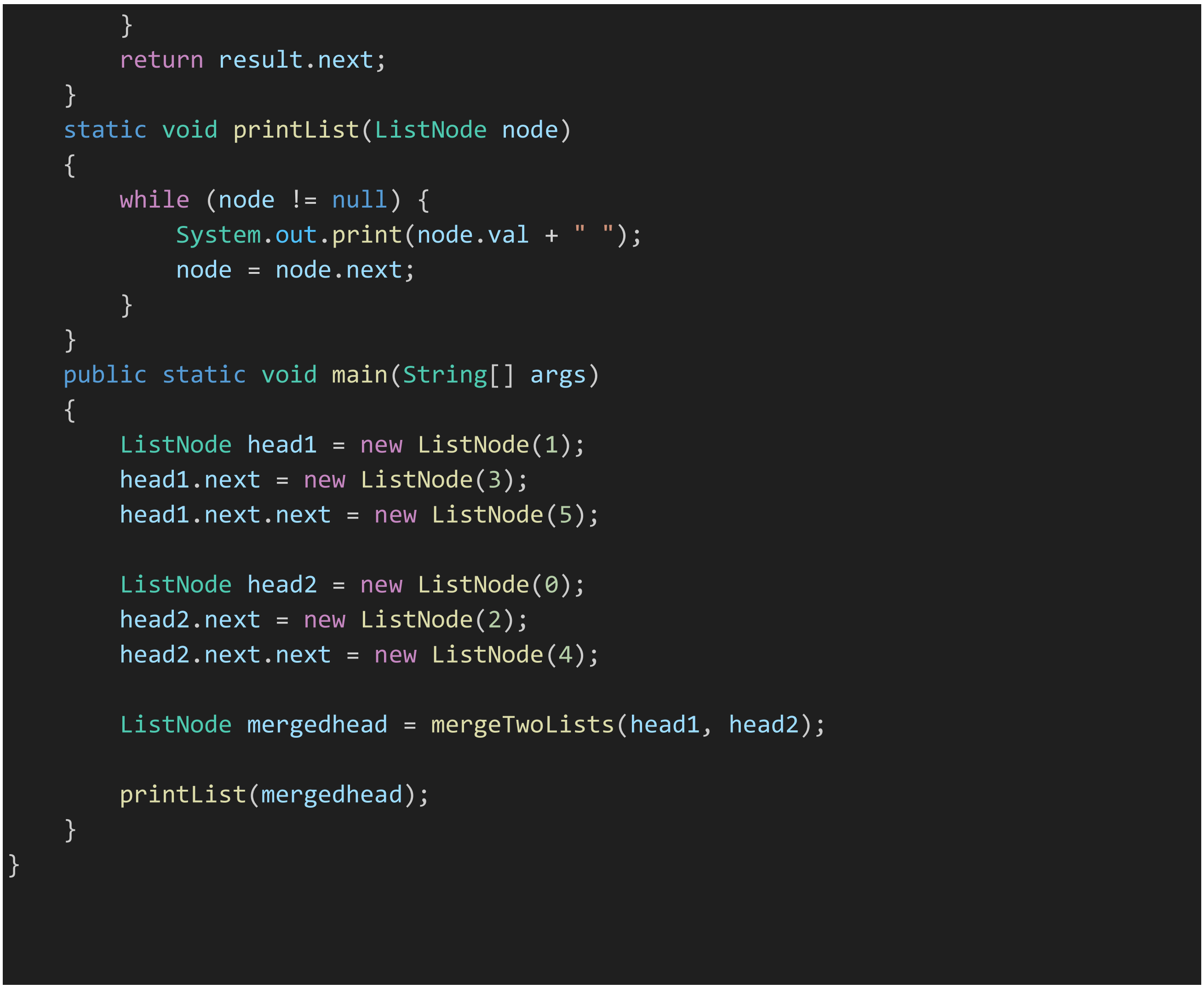
AIM: Write a Program to Merge two linked lists(sorted).

PROBLEM STATEMENT: You are given two sorted linked lists, and you need to merge them into a single sorted linked list.

SOURCE CODE:

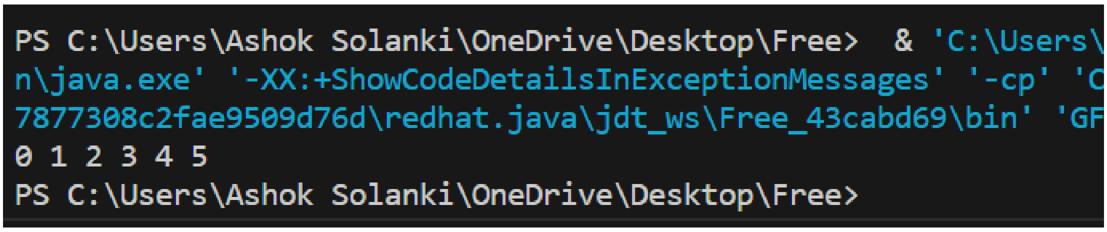


```
public class ListNode {

    int val;
    ListNode next;

    ListNode() {}
    ListNode(int val) { this.val = val; }

    ListNode(int val, ListNode next)
    {
        this.val = val;
        this.next = next;
    }
}
class MergeLinkedlist {
    public static ListNode mergeTwoLists(ListNode l1,ListNode l2){
        ListNode result = new ListNode(-1);
        ListNode p = result;
        while (l1 != null && l2 != null) {

            if (l1.val <= l2.val) {
                p.next = l1;
                l1 = l1.next;
            }
            else {
                p.next = l2;
                l2 = l2.next;
            }
            p = p.next;
        }
        if (l1 == null) {
            p.next = l2;
        }
        else if (l2 == null) {
            p.next = l1;
```



```
        }
        return result.next;
    }
    static void printList(ListNode node)
    {
        while (node != null) {
            System.out.print(node.val + " ");
            node = node.next;
        }
    }
    public static void main(String[] args)
    {
        ListNode head1 = new ListNode(1);
        head1.next = new ListNode(3);
        head1.next.next = new ListNode(5);

        ListNode head2 = new ListNode(0);
        head2.next = new ListNode(2);
        head2.next.next = new ListNode(4);

        ListNode mergedhead = mergeTwoLists(head1, head2);

        printList(mergedhead);
    }
}
```

OUTPUT:



```
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free>  & 'C:\Users\
n\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C
7877308c2fae9509d76d\redhat.java\jdt_ws\Free_43cabd69\bin' 'GF
0 1 2 3 4 5
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free>
```

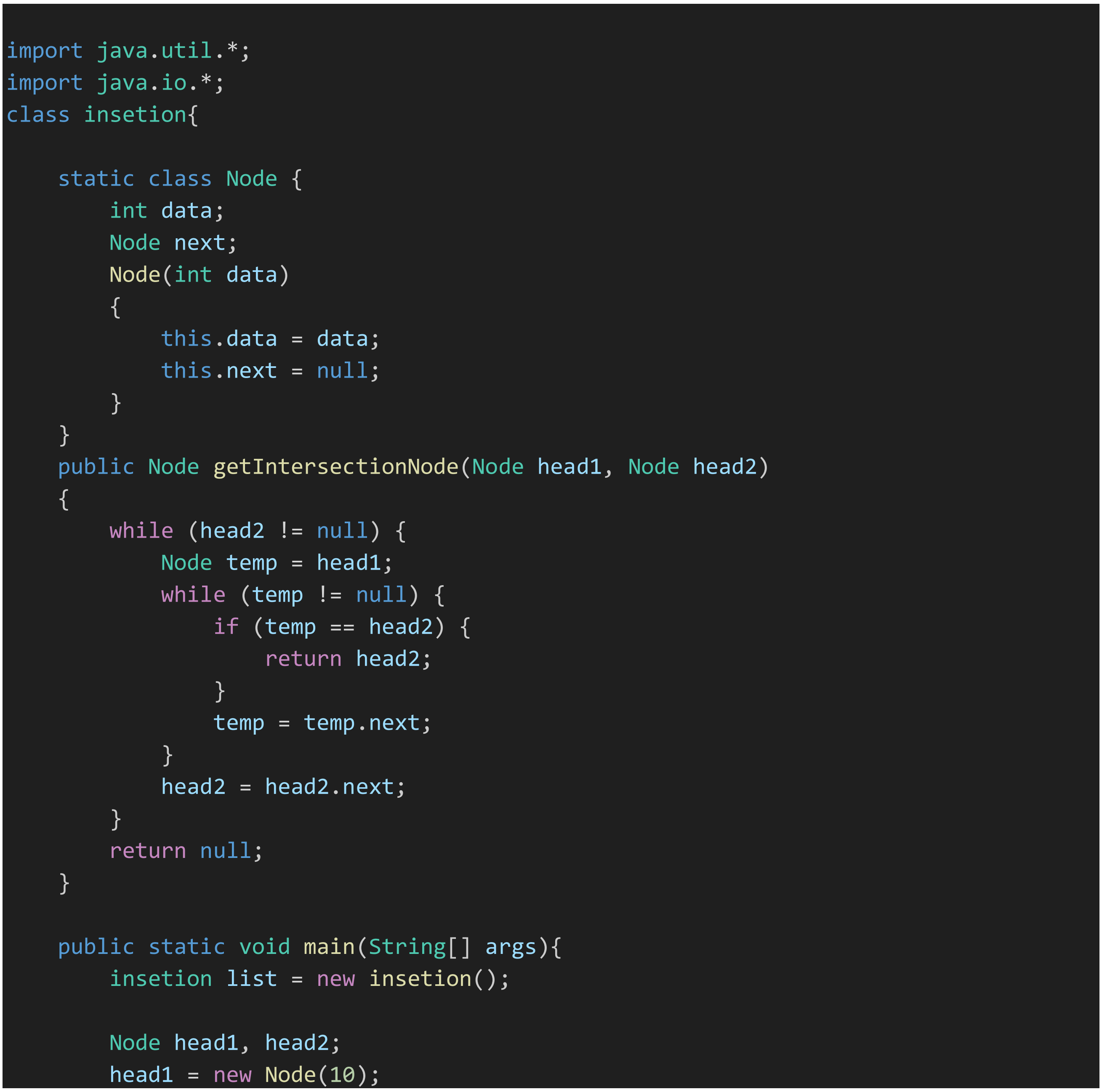
CONCLUSION: The above code is successfully executed in Lab.

PRACTICAL NO: 7

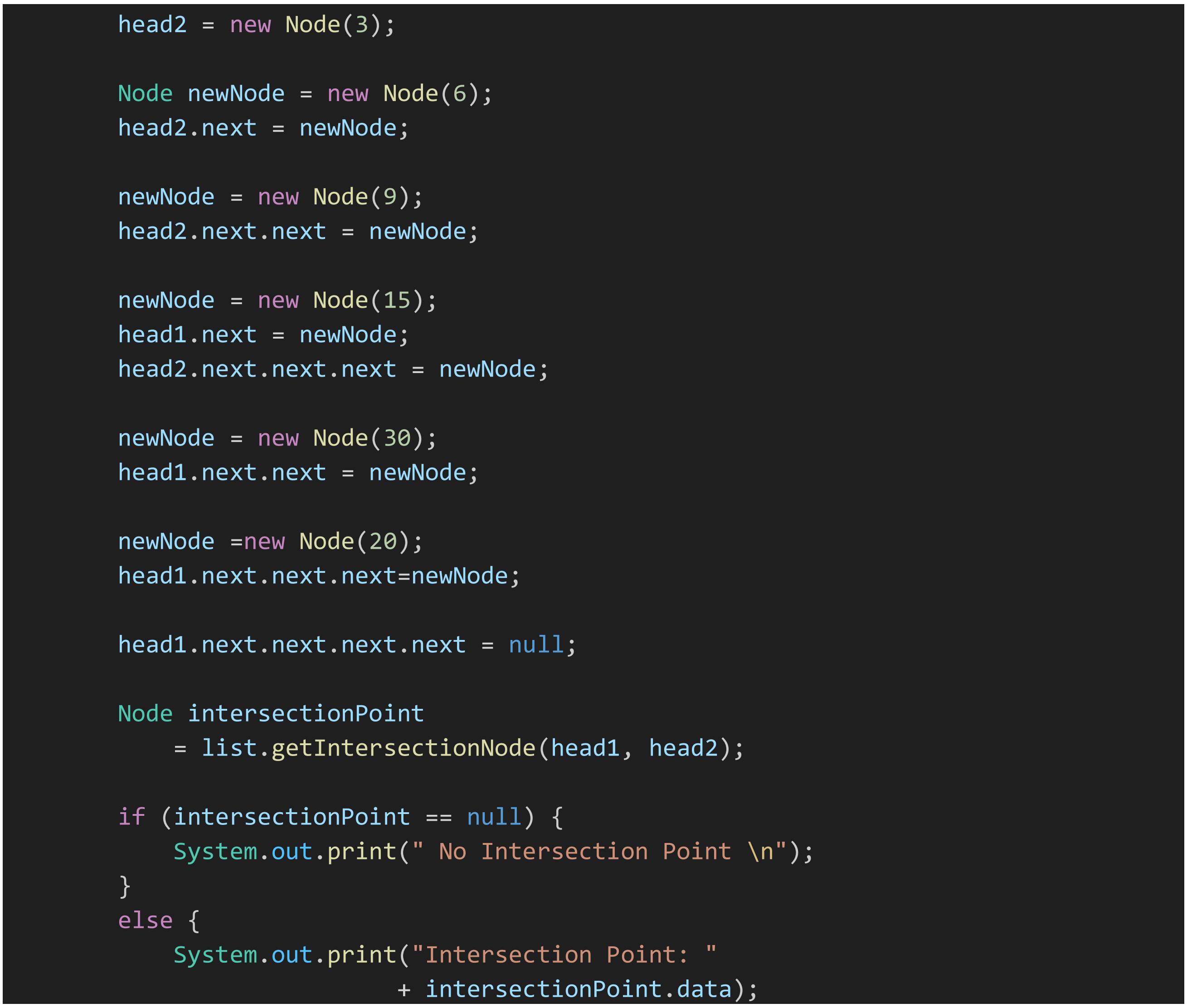
AIM: Write a Program to find the Merge point of two linked lists(sorted)

PROBLEM STATEMENT: You are given two sorted linked lists, list1 and list2. Each linked list is sorted in ascending order.

SOURCE CODE:

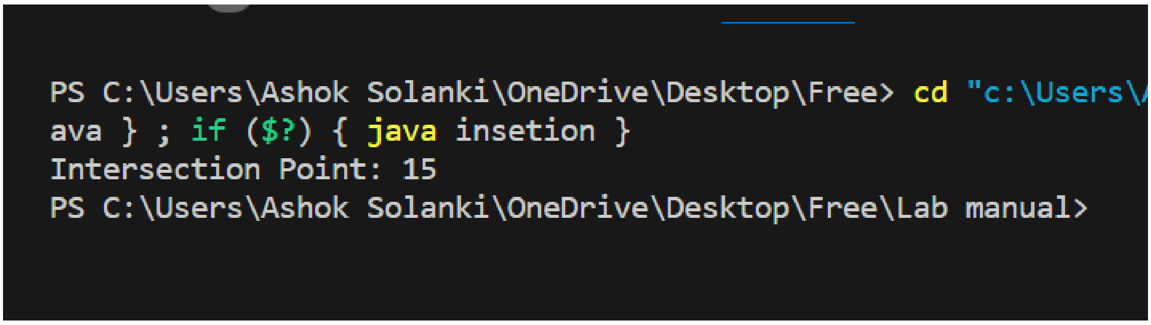


```
import java.util.*;
import java.io.*;
class insetion{

    static class Node {
        int data;
        Node next;
        Node(int data)
        {
            this.data = data;
            this.next = null;
        }
    }
    public Node getIntersectionNode(Node head1, Node head2)
    {
        while (head2 != null) {
            Node temp = head1;
            while (temp != null) {
                if (temp == head2) {
                    return head2;
                }
                temp = temp.next;
            }
            head2 = head2.next;
        }
        return null;
    }

    public static void main(String[] args){
        insetion list = new insetion();

        Node head1, head2;
        head1 = new Node(10);
```



```
        head2 = new Node(3);

        Node newNode = new Node(6);
        head2.next = newNode;

        newNode = new Node(9);
        head2.next.next = newNode;

        newNode = new Node(15);
        head1.next = newNode;
        head2.next.next.next = newNode;

        newNode = new Node(30);
        head1.next.next = newNode;

        newNode =new Node(20);
        head1.next.next.next=newNode;

        head1.next.next.next.next = null;

        Node intersectionPoint
            = list.getIntersectionNode(head1, head2);

        if (intersectionPoint == null) {
            System.out.print(" No Intersection Point \n");
        }
        else {
            System.out.print("Intersection Point: "
                            + intersectionPoint.data);
```

OUTPUT:



```
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free> cd "c:\Users\
ava } ; if ($?) { java insetion }
Intersection Point: 15
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free\Lab manual>
```

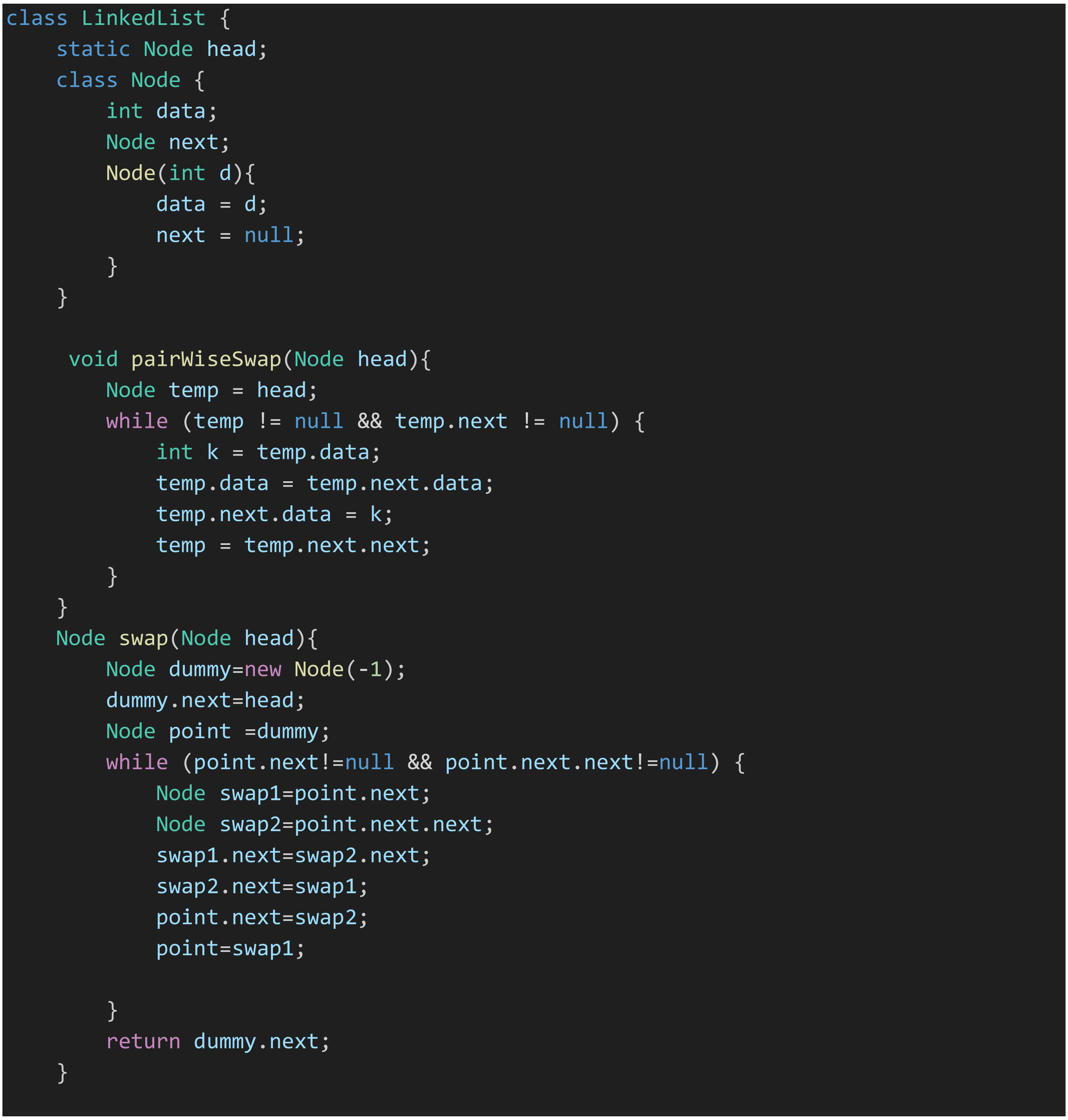
CONCLUSION: The above code is successfully executed in Lab.

PRACTICAL NO: 8

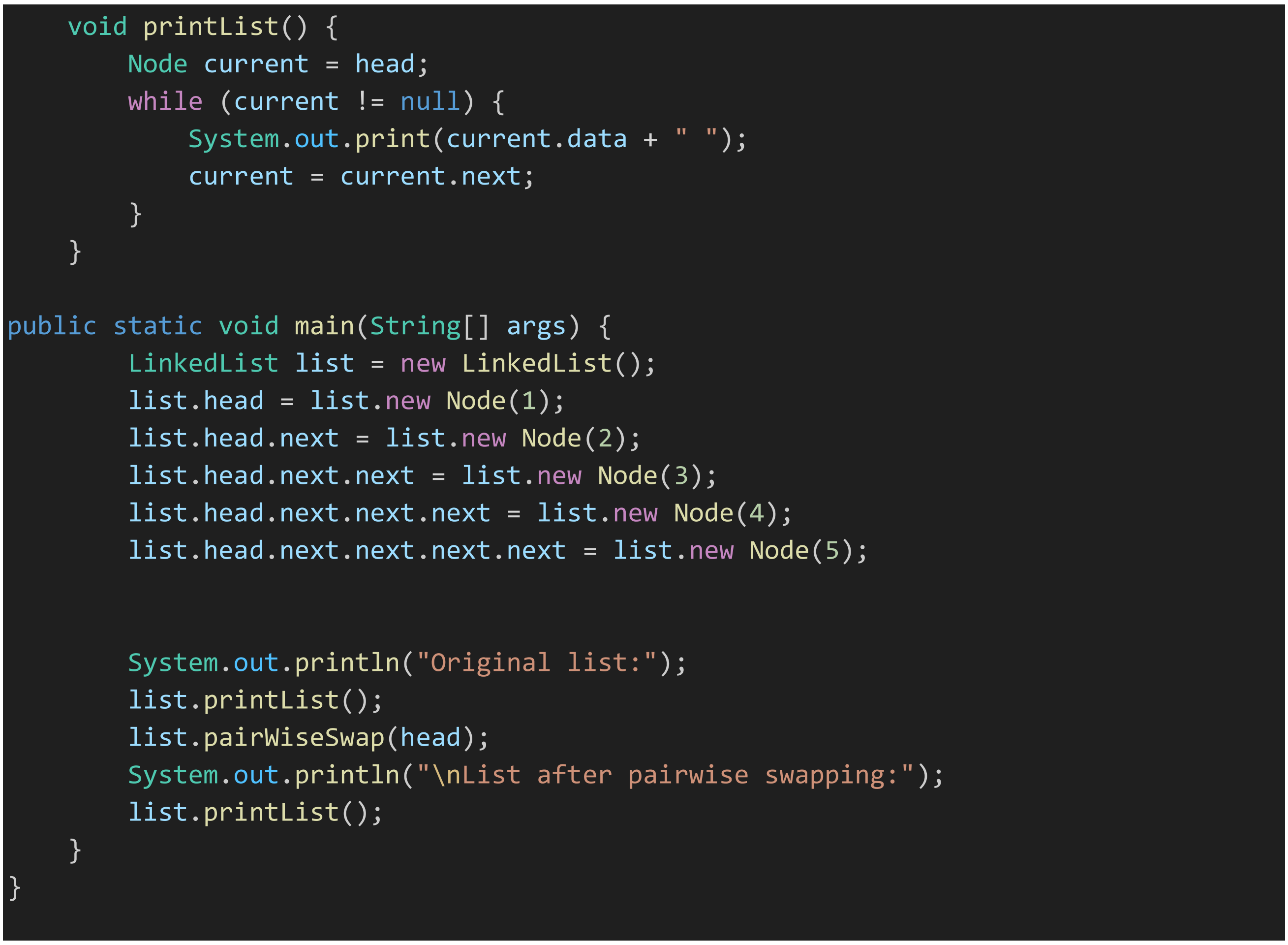
AIM: Write a Program to Swap Nodes pairwise.

PROBLEM STATEMENT: You are given a linked List ,you can swap the node pairwise.

SOURCE CODE:

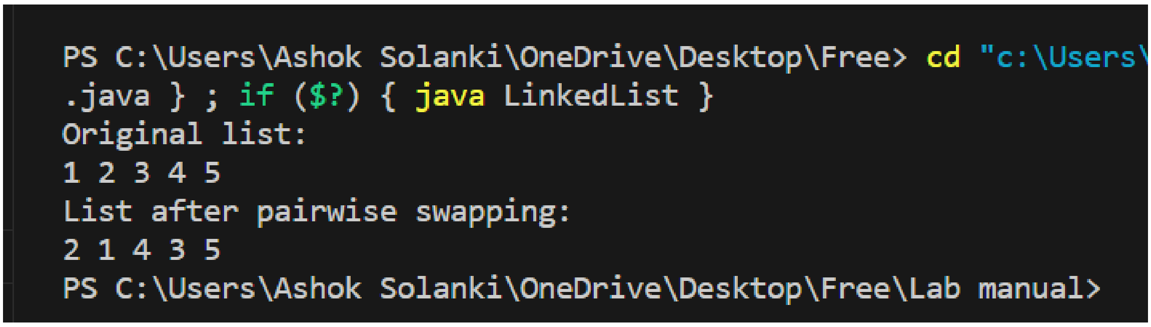


```
class LinkedList {
    static Node head;
    class Node {
        int data;
        Node next;
        Node(int d){
            data = d;
            next = null;
        }
    }

     void pairWiseSwap(Node head){
        Node temp = head;
        while (temp != null && temp.next != null) {
            int k = temp.data;
            temp.data = temp.next.data;
            temp.next.data = k;
            temp = temp.next.next;
        }
    }
    Node swap(Node head){
        Node dummy=new Node(-1);
        dummy.next=head;
        Node point =dummy;
        while (point.next!=null && point.next.next!=null) {
            Node swap1=point.next;
            Node swap2=point.next.next;
            swap1.next=swap2.next;
            swap2.next=swap1;
            point.next=swap2;
            point=swap1;

        }
        return dummy.next;
    }
```



```
    void printList() {
        Node current = head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
    }

public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.head = list.new Node(1);
        list.head.next = list.new Node(2);
        list.head.next.next = list.new Node(3);
        list.head.next.next.next = list.new Node(4);
        list.head.next.next.next.next = list.new Node(5);


        System.out.println("Original list:");
        list.printList();
        list.pairWiseSwap(head);
        System.out.println("\nList after pairwise swapping:");
        list.printList();
    }
}
```

OUTPUT:



```
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free> cd "c:\Users\
.java } ; if ($?) { java LinkedList }
Original list:
1 2 3 4 5
List after pairwise swapping:
2 1 4 3 5
PS C:\Users\Ashok Solanki\OneDrive\Desktop\Free\Lab manual>
```

CONCLUSION: The above code is successfully executed in Lab.