# NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

# OOPS LAB ASSIGNMENT-1

**Name:** P MANOHAR RAO

**Roll No:**197158

**Section:** A

1. Write a program in Java to determine whether a given matrix is a sparse matrix or not

```
import java . util . Scanner ;
public class sparse {
static boolean spar ( int [][] arr , int rows , int columns ) {
int count = 0 ;
for ( int i = 0 ; i < rows ; i ++) {
for ( int j = 0 ; j < columns ; j ++) {
if ( arr [ i ][ j ] == 0 )
count += 1 ;
}
}
return count > (( rows * columns ) / 2 );
}
public static void main ( String [] args ) {
Scanner sc = new Scanner ( System . in );
System . out . print ( "enter no of rows and columns:" );

int rows = sc . nextInt ();
int columns = sc . nextInt ();
int [][] arr = new int [ rows ][ columns ];
System . out . print ( "enter matrix:" );
for ( int i = 0 ; i < rows ; i ++) {
for ( int j = 0 ; j < columns ; j ++) {
arr [ i ][ j ] = sc . nextInt ();
}
}
if ( spar ( arr , rows , columns )) {
System . out . println ( "sparse matrix" );
} else {
System . out . println ( "non sparse matrix" );
}
sc . close ();}}
```

## 2. Write a Java Program to create and display a singly linked list.

```
import java.util.*;

class Node{
    public
    int data;
    Node next;
    public Node(int d) {
        this.data = d;
        this.next = null;
    }
}

public class linkedlist_imp {

    public static void insert( Node head , int  n ){
            Node temp = new Node(n);
            Node Main = head;
            while( Main.next != null ){
                Main = Main.next;
            }
            Main.next = temp;
    }

    public static void print( Node head ){
        Node temp = head;
        while( temp != null ){
            System.out.println( temp.data );
            temp = temp.next;
        }
    }

    public static void main( String args[]){
        Scanner input = new Scanner(System.in);
         int n = input.nextInt();
        Node head = new Node(n);
```

```
            head.next = null;
            n  = input.nextInt();
            while( n != -1 ){
                insert( head , n );
                n = input.nextInt();
            }
          print( head );
        }

      }
```

## 3.Write a Java program to find the maximum and minimum value node from a linked list

```
import java.util.*;

class Node{
    public
    int data;
    Node next;
    public Node(int d) {
        this.data = d;
        this.next = null;
    }
}
class max_and_min_ll{

    public static void insert( Node head , int  n ){
            Node temp = new Node(n);
            Node Main = head;
            while( Main.next != null ){
                Main = Main.next;
            }
            Main.next = temp;
    }

    public static void print( Node head ){
        Node temp = head;
        while( temp != null ){
            System.out.println( temp.data );
            temp = temp.next;
        }
    }

    public static int get_min( Node temp ){
        if( temp == null ) return -1;
        int val = temp.data; temp = temp.next;
    while( temp != null ){
        if( temp.data < val ) val = temp.data;
        temp = temp.next;
    }
    return val;
}

public static int get_max( Node head ){
    if( head == null ) return -1;
    int val = head.data; head = head.next;
    while( head != null ){
        if( head.data > val ) val = head.data;
        head = head.next;
    }
    return val;
}
public static void main( String args[]){
    Scanner input = new Scanner(System.in);
    int n = input.nextInt();
    Node head = new Node(n);
    head.next = null;
    n  = input.nextInt();
    while( n != -1 ){
        insert( head , n );
        n = input.nextInt();
    }
   /*print( head );*/
   Node temp = head;
   System.out.println("The min value is " + get_min( temp ));
   System.out.println("The max value is " + get_max( head ));
}
```

## 4.Write a Java program to delete a node from the middle of the singly linked list import

```
 import java.util.*;

class Node{
    public
    int data;
    Node next;
    public Node(int d) {
        this.data = d;
        this.next = null;
    }
}
```

```
public class DelMidNode{

    public static void insert( Node head , int  n ){
            Node temp = new Node(n);
            Node Main = head;
            while( Main.next != null ){
                Main = Main.next;
            }
            Main.next = temp;
    }

    public static void print( Node head ){
        Node temp = head;
        while( temp != null ){
            System.out.println( temp.data );
            temp = temp.next;
        }
    }

    public static Node Del( Node head ,  Node sptr , Node fptr ){
        while( fptr.next != null ){
            sptr = sptr.next;
            fptr = fptr.next.next;
        }
        return sptr;
    }

    public static void main( String args[]){
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        Node head = new Node(n);
        n  = input.nextInt();
        while( n != -1 ){
            insert( head , n );
            n = input.nextInt();
        }

        Node sptr,fptr;
        sptr = head;  fptr = head.next.next;
        Node temp = Del( head , sptr , fptr  );
        Node Main = temp;
        while( Main != null ){
           if( Main.next == temp.next ){
               Node dummy = Main.next.next;
               Main.next = dummy;
               break;
           }
        }

        print( head );
    }

 }
```

5. Write a program in Java to implement multilevel inheritance which shows the usage of super and final keyword.

```
public class inheritace {
class a {
String name = "class A" ;
void print () {
System . out . println ( "A" );
}
}
class b extends a {
final String name = "class B" ;
final void print () {
System . out . println ( "B" );
super . print ();
}
}
final class c extends b {
String name = "class C" ;
}
public static void main ( String [] args ) {
inheritace i = new inheritace ();
inheritace . c temp = i . new c ();
System . out . println ( temp . name );
temp . print ();
}
}
```

6. Write a java program to create two arrays(Unsorted) then sort them by using the best sorting algorithm (Recommended quick sort) then merge these two

arrays.

Testcases:

1) array 1: 2 9 7 11 12

Array 2: 8 10 3 1 16

Output: 1 2 3 7 8 9 10 11 12 16

```
 import java.util.*;

public class Mrgtwosort {
    public static void swap( int A[] , int a , int b )
    {
        int temp = A[a];
        A[a] = A[b];
        A[b] = temp;
    }
    public static int Partition( int A[] , int l , int h )
    {
        int t = h-1;
        int P = A[t];
        while( l < h )
        {
            while( A[l] < P ) l++;
            while( A[h] > P ) h--;
            if( l < h ) swap( A , l , h  );
        }
        if( l < h )
        swap( A , t , h );
        return h;
    }
    public static void QuickSort( int A[] , int l , int h )
    {
        if( l < h )
        {
            int j = Partition( A , l , h );
            QuickSort( A , l , j-1 );
            QuickSort( A , j+1 , h );
        }
    }
    public static void mrgtwoarr(int[] arr1, int[] arr2, int n1,  int n2, int[] arr3)
    {
        int i = 0, j = 0, k = 0;
        while ( i<n1 && j <n2 )
        {
            if (arr1[i] < arr2[j])     arr3[k++] = arr1[i++];
            else   arr3[k++] = arr2[j++];
        }
        while ( i < n1 )  arr3[k++] = arr1[i++];
        while (j < n2 )    arr3[k++] = arr2[j++];
    }
    public static void main( String args[] ){
        Scanner input = new Scanner(System.in);

        System.out.println("Enter the size of the array one ");
        int n1 = input.nextInt();
        int arr1[] = new int[n1];
        System.out.println("Enter " + n1 +" elements for array one " );
        for( int i = 0 ; i < n1; i++ ) arr1[i] = input.nextInt();

        System.out.println("Enter the size of the array 2 ");
        int n2 = input.nextInt();
        int arr2[] = new int[n2];
        System.out.println("Enter " + n2 +" elements for array two " );
        for( int i = 0 ; i < n2; i++ ) arr2[i] = input.nextInt();

        QuickSort( arr1 , 0 , n1-1 );
        QuickSort( arr2 , 0 , n2-1 );
        int arr3[] = new int[n1+n2];
        mrgtwoarr(arr1, arr2, n1, n2, arr3);
        System.out.println("Array after merging");
        for (int i=0; i < n1+n2; i++)
            System.out.print(arr3[i] + " ");
        }
}
```

7. Write a java program to left and right rotate the array by a given number of positions.

(Number of positions to shift must be read from the console.)

Testcases:

1) Array: 9 5 7 2 6 3 8

Positions: 3

```
Output:
Left rorate: 2 6 3 8 9 5 7
Right rotate: 6 3 8 9 5 7 2
```

```
import java . util . Scanner ;
public class rotate {
static void left ( int [] arr , int post ) {
int [] temp = new int [ arr . length ];
for ( int i = post ; i < arr . length ; i ++) {
temp [ i - post ] = arr [ i ];
}
for ( int i = arr . length - post ; i < arr . length ; i ++) {
temp [ i ] = arr [ i + post - arr . length ];
}
for ( int i = 0 ; i < temp . length ; i ++) {
System . out . print ( temp [ i ] + " " );
}
System . out . println ();
}
static void right ( int [] arr , int post ) {
int [] temp = new int [ arr . length ];
for ( int i = 0 ; i < post ; i ++) {
temp [ i ] = arr [ arr . length - post + i ];
}
for ( int i = post ; i < arr . length ; i ++) {
temp [ i ] = arr [ i - post ];
}
for ( int i = 0 ; i < temp . length ; i ++) {
System . out . print ( temp [ i ] + " " );
}
System . out . println ();
}
public static void main ( String [] args ) {
Scanner sc = new Scanner ( System . in );
int size = sc . nextInt ();
int [] arr = new int [ size ];
for ( int i = 0 ; i < size ; i ++) {
arr [ i ] = sc . nextInt ();
}
int rot = sc . nextInt ();
left ( arr , rot );
right ( arr , rot );
sc . close ();
}
}
```

8.Given an unsorted array of elements, find the longest consecutive elements sequence (must be in ascending order) in the array using a Java program. Testcases:

Array: 1 8 3 7 2 4 8 1

Output: 2 4 8

```
import java.util.*;

public class LIS {

   public static int[] LongSubSeq( int arr[] , int dp[] , int n ){
      for( int i = 1 ; i < n ; i ++ ){
         for( int j  = 0 ; j < i ; j++ ){
            if( arr[i] > arr[j] && dp[i] < dp[j] + 1){
               dp[i] = dp[j] + 1;
            }
         }
      }
      return dp;
   }

   public static void main( String args[] ){
      System.out.println("Hey, Enter the size of array ");
      Scanner input = new Scanner(System.in);
      int n = input.nextInt();

      //input
      int arr[] = new int[n];
      for( int i = 0 ; i < n ; i ++ ){
         arr[i] = input.nextInt();
      }

      int dp[] = new int[n];
      Arrays.fill(dp, 1);
```

```
        int d[] = LongSubSeq( arr , dp , n );
        int max = 0 , k = 0 ;
        for( int i = 0 ; i < n ; i++ ){

            if( d[k] > max ){
                System.out.println(arr[k]);
                max = d[k];
                k += 1;
            }
            else{
                k += 1;
            }
        }
    }

}
```

9. Write a java program to create a 2D array and write code to print the spiral traversal of that

array? Take array length and read array from the console.

Ex:

1 2 3 4

5 6 7 8 output 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

9 10 11 12

13 14 15 16

```
import java . util . Scanner ;
public class array2d {
public static void main ( String [] args ) {
Scanner sc = new Scanner ( System . in );
int row , column ;
System . out . print ( "enter rows and columns:" );
row = sc . nextInt ();
column = sc . nextInt ();
int [][] arr = new int [ row ][ column ];
for ( int i = 0 ; i < row ; i ++) {
for ( int j = 0 ; j < column ; j ++) {
arr [ i ][ j ] = sc . nextInt ();
}
}
for ( int i = 0 ; i < row ; i ++) {
```

```
for ( int j = 0 ; j < column ; j ++) {
System . out . print ( arr [ i ][ j ] + " " );
}
}
sc . close ();
}
}
```

## 10.Given an array, write a java program to find an index of the smallest element is to be removed such that array elements sum will be divisible by k.

Testcases:

1. Array: 4 12 17 24 8 ,     k=8

   Output: 2

```
import java.util.*;

public class SmlNumRem{
   public static void main( String[] args ){

      System.out.println("Enter the size of array ");
      Scanner input = new Scanner(System.in);

      //size of array
      int n = input.nextInt();
      int A[] = new int[n];

      //array input
      System.out.println("Enter the " + n +" elements of array ");
      for( int i = 0 ; i < n ; i ++ ){
         A[i] = input.nextInt();
      }

      System.out.println("Enter the k value ");
      int k = input.nextInt();

      int sum = 0;
      for( int i = 0 ; i < n ; i++ ){
         sum += A[i];
      }

      int val = 0 ,loc = 0,temp = A[0];
```

```
        if( (sum - A[0]) % k == 0 ){
            val = A[0];
            loc = 0;
        }

        for( int i = 1 ; i < n ; i++ ){
            if( ((sum-A[i])%k) == 0 ){
                if( val > A[i] ) val = A[i]; loc = i;
            }
        }

        System.out.println("The location is " + loc);
    }
}
```

11. Create a class called as Company which is a base class with name and salary as fields and

take 5 employee details as input from the user. Derive one class containing methods

dispMin() and dispMax() to display the employee name with minimum and maximum salary. Derive another class containing the methods avgSalary() and difference() which displays the

average salary of employee and the difference between maximum and minimum salary. Write

a Java program to implement the same.

Test Case:

Input: Name salary

John 10,000

Mike 21,300

Bruce 18,700

Michael 56,936

Julie 41, 749

Output:

Min Salary: 10,000

Max Salary: 56,936

Avg. Salary: 29,737

Difference: 46,936

```
import java . util .*;
public class company {
class subcompany {
int [] salary ;
String [] names ;
int count ;
public subcompany ( int size ) {
salary = new int [ size ];
names = new String [ size ];
count = 0 ;
}
public void push ( String name , int salary ) {
this . names [ count ] = name ;
this . salary [ count ] = salary ;
count += 1 ;
}
}
class disp extends subcompany {
public int dispMin () {
int min = Integer . MAX_VALUE ;
```

12. Create a base class Shape containing name as field. Class Shape will have a public method called getName() that returns the name of the shape. Create a class Circle deriving Shape having radius as field and calculate() method to calculate the area. Then, create a class Cylinder deriving Circle having height as field and calculate() method. Take the radius and height values from user as input and calculate areas of circle and cylinder as output. Write a Java program to implement the same.

```
 import java.util.*;

class shape
{
   String name;
   Scanner sc=new Scanner(System.in);

    shape()
    {
        System.out.println("Enter the name");        this.name=sc.next();
        }
        public String getName()
        {
            return this.name;
        }
    }
    class circle extends shape
    {
        float radius;
        Scanner input=new Scanner(System.in);     circle()
        {
             System.out.println("Enter the radius");
            this.radius=input.nextInt();
        }
        public double calculate()
        {
            return Math.PI*this.radius*this.radius;
        }
    }
    class cylinder extends circle
    {
        int height;
        Scanner input=new Scanner(System.in);
    cylinder() { super();
            System.out.println("Enter the height ");         this.height = input.nextInt() ;
        }
        public double calculate()
        {
            double area1=2*super.calculate();
            System.out.println("area of circle "+area1/2);        double area2=2*Math.PI*this.radius*this.height;
    return area1+area2;
        }
    }
    public class areas_shapes {
        public static void main(String[] args)
        {
            Scanner    input=new    Scanner(System.in);                              cylinder    obj=new    cylinder();
    System.out.println(obj.calculate());
        }
}
```