Maharshi Dayanand University

University Institute of Engineering and Technology

Programming in Java (JAVA)

Assignment - 1

Submitted By:

Ujjwal Kumar Singh

27540

CSE - A 5th Sem

Submitted To:

Mr. Dheeraj Sahni

Java Basic Programs

1) Write a program for Fibonacci Series in Java.

```
import java.util.Scanner;
public class JavaExample {

    public static void main(String[] args) {

        int count, num1 = 0, num2 = 1;
        System.out.println("How many numbers you want in the sequence:");
        Scanner scanner = new Scanner(System.in);
        count = scanner.nextInt();
        scanner.close();
        System.out.print("Fibonacci Series of "+count+" numbers:");

        int i=1;
        while(i<=count)
        {
            System.out.print(num1+" ");
            int sumOfPrevTwo = num1 + num2;
            num1 = num2;
            num2 = sumOfPrevTwo;
            i++;
        }
    }
}
```

```
How may numbers you want in the sequence:
11
Fibonacci Series of 11 numbers:0 1 1 2 3 5 8 13 21 34 55
```

2) Write a program for Prime Number Program in Java

```
import java.util.Scanner;
public class PrimeCheck
{
   public static void main (String args[])
   {
            int temp;
            boolean isPrime=true;
            Scanner scan= new Scanner(System.in);
```

```
        System.out.println("Enter any number:");
        int num=scan.nextInt();
     scan.close();
        for(int i=2;i<=num/2;i++)
        {
       temp=num%i;
          if(temp==0)
          {
            isPrime=false;
            break;
          }
        }
        //If isPrime is true then the number is prime else not
        if(isPrime)
          System.out.println(num + " is a Prime Number");
        else
          System.out.println(num + " is not a Prime Number");
  }
}
```

```
Enter any number:
13
13 is a Prime Number
```

```
Enter any number:
23507
23507 is not a Prime Number
```

3) Write a program for Palindrome Program in Java

```
import java.util.*;
public class Main
{
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the number: ");
      String reverse = "";
      String num = sc.nextLine();
      int length = num.length();
     for ( int i = length - 1; i >= 0; i-- )
        reverse = reverse + num.charAt(i);
     if (num.equals(reverse))
        System.out.println("The entered string " +num +" is a palindrome.");
     else
        System.out.println("The entered string " +num +"  isn't a palindrome.");
  }
```

}

```
Enter the number:
16461
The entered string 16461 is a palindrome.
```

```
Enter the number:
23507
The entered string 23507  isn't a palindrome.
```

4) Write a program for Factorial Program in Java

```
import java.util.*;
public class Main
{
    public static void main(String []args)
    {
       Scanner sc=new Scanner(System.in);
       System.out.println("Enter the number: ");
       int num=sc.nextInt();
       int i=1,fact=1;
       while(i<=num)
       {
          fact=fact*i;
          i++;
       }
       System.out.println("Factorial of the number: "+fact);
    }
}
```

```
Enter the number:
7
Factorial of the number: 5040
```

5) Write a program for Armstrong Number in Java

```
import java.util.Scanner;
public class ArmStrong
{
   public static void main(String[] args)
   {
      int n, count = 0, a, b, c, sum = 0;
      Scanner s = new Scanner(System.in);
```

```
        System.out.print("Enter any integer you want to check:");
        n = s.nextInt();
        a = n;
        c = n;
        while(a > 0)
        {
            a = a / 10;
            count++;
        }
        while(n > 0)
        {
            b = n % 10;
            sum = (int) (sum+Math.pow(b, count));
            n = n / 10;
        }
        if(sum == c)
        {
            System.out.println("Given number is Armstrong");
        }
        else
        {
            System.out.println("Given number is not Armstrong");
        }
    }
}
```

```
Enter any integer you want to check:153
Given number is Armstrong
```

```
Enter any integer you want to check:23507
Given number is not Armstrong
```

6) Write a program to Generate Random Number in Java

```
import java.lang.Math;

public class RandomNumber
{
public static void main(String args[])
{
System.out.println("Random Number: " + Math.random());
}
}
```

```
Random Number: 0.90215092056564
```

7) Write a program to Print Pattern in Java

```
public class RightTrianglePattern
{
public static void main(String args[])
{
//i for rows and j for columns
int i, j, row=6;
//outer loop for rows
for(i=0; i<row; i++)
{
//inner loop for columns
for(j=0; j<=i; j++)
{
//prints stars
System.out.print("* ");
}
System.out.println();
}
}
}
```

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

8) Write a program to Compare Two Objects in Java

```
import java.util.Scanner;
public class Compare {
 public static void main( String args[] )
   {

      Scanner input = new Scanner(System.in);
      int number1;
      int number2;
```

```
      System.out.print( "Input first integer: " );
      number1 = input.nextInt();

      System.out.print( "Input second integer: " );
      number2 = input.nextInt();

      if ( number1 == number2 )
         System.out.printf( "%d == %d\n", number1, number2 );
      if ( number1 != number2 )
         System.out.printf( "%d != %d\n", number1, number2 );
      if ( number1 < number2 )
         System.out.printf( "%d < %d\n", number1, number2 );
      if ( number1 > number2 )
         System.out.printf( "%d > %d\n", number1, number2 );
      if ( number1 <= number2 )
         System.out.printf( "%d <= %d\n", number1, number2 );
      if ( number1 >= number2 )
         System.out.printf( "%d >= %d\n", number1, number2 );
   }
}
```

```
Input first integer: 23507
Input second integer: 22222
23507 != 22222
23507 > 22222
23507 >= 22222
```

10) Write a program to Print ASCII Value in Java

```
import java.util.*;
public class Main
{
    public static void main(String []args)
    {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter the character whose ASCII value you want to know ");
      char ch=sc.next().charAt(0);
      int value=ch;
      System.out.println("Character is "+ch+" and ASCII value is "+value);
    }
}
```

```
Enter the character whose ASCII value you want to know
R
Character is R and ASCII value is 82
```

```
Enter the character whose ASCII value you want to know
r
Character is r and ASCII value is 114
```

Java Array Programs

1) Java Program to copy all elements of one array into another array

```
import java.util.Scanner;
import java.util.Arrays;

public class CopyArray1 {
	private static Scanner sc;
	public static void main(String[] args)
	{
		int Size, i;
		sc = new Scanner(System.in);

		System.out.print(" Please Enter Number of elements in an array : ");
		Size = sc.nextInt();

		int [] a = new int[Size];

		System.out.print(" Please Enter " + Size + " elements of an Array  : ");
		for (i = 0; i < Size; i++)
		{
			a[i] = sc.nextInt();
		}

		int[] b = Arrays.copyOf(a, Size);

		System.out.println("\n **** Elements in b[] Array after Copying are **** ");
		for (i = 0; i < Size; i++)
		{
			System.out.println(" Element at b["+ i +"] = " + b[i]);
		}
	}
}
```

```
Please Enter Number of elements in an array : 5
Please Enter 5 elements of an Array  : 2 3 5 0 7

 **** Elements in b[] Array after Copying are ****
Element at b[0] = 2
Element at b[1] = 3
Element at b[2] = 5
Element at b[3] = 0
Element at b[4] = 7
```

2) Java Program to find the frequency of each element in the array

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
int[] arr = new int[100];
int[] freq = new int[100];
int size, i, j, count;
/* Input size of array */
System.out.print("Enter size of array: ");
size = sc.nextInt();
/* Input elements in array */
System.out.print("Enter elements in array: ");
for(i=0; i<size; i++)
{
arr[i] = sc.nextInt();
/* Initially initialize frequencies to -1 */
freq[i] = -1;
}
for(i=0; i<size; i++)
{
count = 1;
for(j=i+1; j<size; j++)
{
/* If duplicate element is found */
if(arr[i]==arr[j])
{
count++;
freq[j] = 0;
}
}
/* If frequency of current element is not counted */
if(freq[i] != 0)
{
freq[i] = count;
}
}
/* Print frequency of each element */
System.out.print("Frequency of all elements of array : ");
for(i=0; i<size; i++)
{
if(freq[i] != 0)
{
System.out.print(arr[i] + " occurs " + freq[i] + " times" + "\n");
}
```

```
}
}
}
```

```
Enter size of array: 9
Enter elements in array: 1 2 3 4 5 1 2 3 3
Frequency of all elements of array : 1 occurs 2 times
2 occurs 2 times
3 occurs 3 times
4 occurs 1 times
5 occurs 1 times
```

3) Java Program to left rotate the elements of an array

```
class RotateLeft {
    public static void main(String[] args) {
        int [] arr = new int [] { 2, 3, 5, 0, 7};
        //n determine the number of times an array should be rotated
        int n = 3;
        System.out.println("Original array: ");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
        //Rotate the given array by n times toward left
        for(int i = 0; i < n; i++){
            int j, first;
            //Stores the first element of the array
            first = arr[0];
            for(j = 0; j < arr.length-1; j++){
                //Shift element of array by one
                arr[j] = arr[j+1];
            }
            arr[j] = first;
        }
        System.out.println();
        System.out.println("Array after left rotation: ");
        for(int i = 0; i< arr.length; i++){
            System.out.print(arr[i] + " ");
        }
    }
}
```

```
Original array:
2 3 5 0 7
Array after left rotation:
0 7 2 3 5
```

4) Java Program to print the duplicate elements of an array

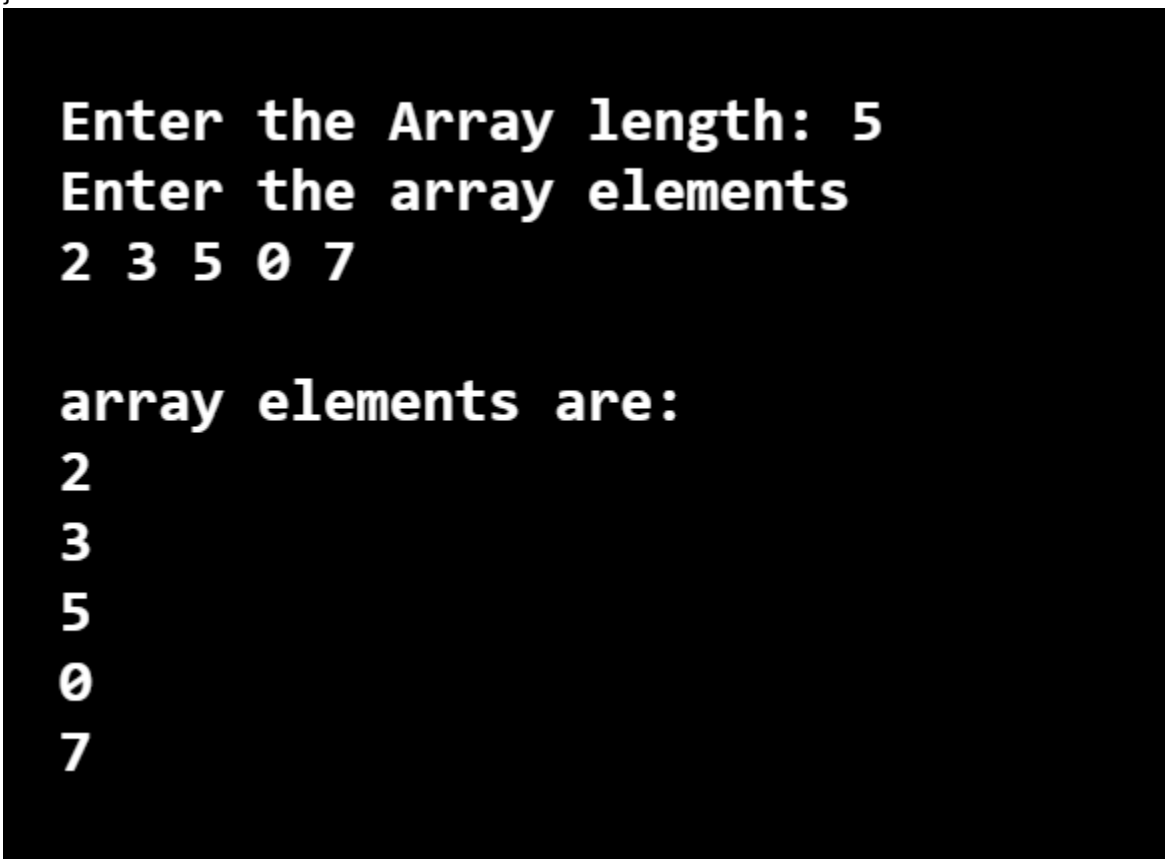
```
public class DuplicateElement {
   public static void main(String[] args) {

      int [] arr = new int [] {1, 2, 3, 4, 2, 7, 8, 8, 3};

      System.out.println("Duplicate elements in given array: ");

      for(int i = 0; i < arr.length; i++) {
         for(int j = i + 1; j < arr.length; j++) {
            if(arr[i] == arr[j])
               System.out.println(arr[j]);
         }
      }
   }
}
```

```
Duplicate elements in given array:
2
3
8
```

5) Java Program to print the elements of an array

```
import java.util.Scanner;
public class Array_Single_Dim{
public static void main (String args[]){
   Scanner scan=new Scanner(System.in);
System.out.print("Enter the Array length: ");
int len=scan.nextInt();
int arr[]=new int[len]; //declaration and creation i dim array
```

```
System.out.println("Enter the array elements");
for(int i=0; i<len; i++){
arr[i]=scan.nextInt();
}
System.out.println("\narray elements are:");
for(int i=0; i<len; i++){
System.out.println(arr[i]);
}//display every array element
}
}
```



6) Java Program to print the elements of an array in reverse order

```
import java.util.Scanner;

public class REV
{
   public static void main(String[] args)
   {
      int n, res,i,j=0;
      Scanner s = new Scanner(System.in);
      System.out.print("Enter number of elements in the array:");
      n = s.nextInt();
      int array[] = new int[n];
```

```
        int rev[] = new int[n];
        System.out.println("Enter "+n+" elements ");
        for( i=0; i < n; i++)
        {
            array[i] = s.nextInt();
        }
        System.out.println("Reverse of an array is :");
        for( i=n;i>0 ; i--,j++)
        {
            rev[j] = array[i-1];
            System.out.println(rev[j]);

        }
    }
}
```

```
Enter number of elements in the array:5
Enter 5 elements
2 3 5 0 7
Reverse of an array is :
7
0
5
3
2
```

7) Java Program to print the elements of an array present on even position

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

        //Initialize array
        int [] arr = new int [] {1, 2, 3, 4, 5};

        System.out.println("Elements of given array present on even position: ");
        //Loop through the array by incrementing value of i by 2
        //Here, i will start from 1 as first even positioned element is present at position 1.
        for (int i = 1; i < arr.length; i = i+2) {
            System.out.println(arr[i]);
        }
    }
}
```

```
Elements of given array present on even position:
2
4
```

8) Java Program to print the elements of an array present on odd position

```
public class OddPosition {
    public static void main(String[] args) {
        //Initialize array
       int [] arr = new int [] {1, 2, 3, 4, 5};
        System.out.println("Elements of given array present on odd position: ");
        //Loop through the array by incrementing value of i by 2
        for (int i = 0; i < arr.length; i = i+2) {
            System.out.println(arr[i]);
        }
    }
}
```

```
Elements of given array present on odd position:
1
3
5
```

9) Java Program to print the largest element in an array

```
import java.util.Scanner;

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

        int arr[]=new int[n]; /
       System.out.println("Enter the array");
       for(int i=0;i<n;i++)
       {
           arr[i]=sc.nextInt();
       }
```

```
        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                if(arr[i]<arr[j])
                {
                    int temp=arr[i];
                    arr[i]=arr[j];
                    arr[j]=temp;
                }
            }
        }

        System.out.println("Largest element is "+arr[0]);

    }
}
```

```
Enter the size of the array
5
Enter the array
0034 344 3554 376 030
Largest element is 3554
```

10) Java Program to print the smallest element in an array

```
import java.util.Scanner;
public class Smallest
{
   public static void main(String[] args)
   {
      int t, i, small;
      Scanner scan = new Scanner(System.in);

      System.out.print("Enter the Size of Array: ");
      t = scan.nextInt();
      int[] arr = new int[tot];
      System.out.print("Enter " +t+ " Numbers: ");
      for(i=0; i<t; i++)
         arr[i] = scan.nextInt();

      small = arr[0];
      for(i=1; i<t; i++)
      {
         if(small>arr[i])
            small = arr[i];
```

```
    }

    System.out.println("\nSmallest Number = " +small);
  }
}
```

```
Enter the Size of Array: 5
Enter 5 Numbers: 33 44 08 097 002

Smallest Number = 2
```

11) Java Program to print the number of elements present in an array

```
public class CountArray {
   public static void main(String[] args) {
     //Initialize array
       int [] arr = new int [] {1, 2, 3, 4, 5};
     //Number of elements present in an array can be found using the length
       System.out.println("Number of elements present in given array: " + arr.length);
   }
}
```

```
Number of elements present in given array: 5
```

12) Java Program to print the sum of all the items of the array

```
import java.util.*;
import java.util.Arrays;

public class Main
{
  public static void main(String args[])
  {
     Scanner sc=new Scanner(System.in);

    int n;
    System.out.println("Enter the total number of elements ");
    n=sc.nextInt();
```

```
    int arr[]=new int[n];
    System.out.println("Enter the elements of the array ");
    for(int i=0; i<n ;i++)
    {
       arr[i]=sc.nextInt();
    }
    int sum = 0;
    for( int num : arr)
    {
       sum = sum+num;
    }

      System.out.println("The sum of all the elements in the array is "+sum);

  }
}
```

```
Enter the total number of elements
5
Enter the elements of the array
2 3 5 0 7
The sum of all the elements in the array is 17
```

13) Java Program to right rotate the elements of an array

```
public class RotateRight {

public static void main(String[] args) {
      int [] arr = new int [] {1, 2, 3, 4, 5};
      //n determine the number of times an array should be rotated.
      int n = 3;

      System.out.println("Original array: ");
      for (int i = 0; i < arr.length; i++) {
         System.out.print(arr[i] + " ");
      }

      for(int i = 0; i < n; i++){
         int j, last;
         last = arr[arr.length-1];

         for(j = arr.length-1; j > 0; j--){
            arr[j] = arr[j-1];
         }
```

```
            arr[0] = last;
        }

        System.out.println();

        System.out.println("Array after right rotation: ");
        for(int i = 0; i< arr.length; i++){
            System.out.print(arr[i] + " ");
        }
    }
}
```

```
Original array:
1 2 3 4 5
Array after right rotation:
3 4 5 1 2
```

14) Java Program to sort the elements of an array in ascending order

```
import java.util.Arrays;
import java.util.Scanner;
import java.util.Collections;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n;
        System.out.println("Enter the number of elements :");
        n=sc.nextInt();

        Integer arr[]=new Integer[n];
        System.out.println("Enter the elements of the array :");
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }

        int temp = 0;

         for (int i = 0; i < arr.length; i++)
         {
            for (int j = i+1; j < arr.length; j++)
            {
               if(arr[i] > arr[j])
               {
```

```
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

    System.out.println();

    System.out.println("Elements of array sorted in ascending order: ");
    for (int i = 0; i < arr.length; i++)
    {
        System.out.print(arr[i] + " ");
    }
}
```

```
Enter the number of elements :
9
Enter the elements of the array :
2 3 5 0 7 9 1 8 4

Elements of array sorted in ascending order:
0 1 2 3 4 5 7 8 9
```

15) Java Program to sort the elements of an array in descending order

```
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n;
        System.out.println("Enter the number of elements :");
        n=sc.nextInt();

        Integer arr[]=new Integer[n];
        System.out.println("Enter the elements of the array :");
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }

        int temp = 0;
```

```
        for (int i = 0; i < arr.length; i++)
        {
          for (int j = i+1; j < arr.length; j++)
          {
            if(arr[i] < arr[j])
            {
               temp = arr[i];
               arr[i] =  arr[j];
               arr[j] = temp;
            }
          }
        }

        System.out.println();

        System.out.println("Elements of array sorted in descending order: ");
        for (int i = 0; i < arr.length; i++)
        {
           System.out.print(arr[i] + " ");
        }
    }
}
}
```

```
Enter the number of elements :
8
Enter the elements of the array :
2 3 5 0 7 1 9 4

Elements of array sorted in descending order:
9 7 5 4 3 2 1 0
```

16) Java Program to Find 3rd Largest Number in an array

```
import java.util.Scanner;
public class FindElement
{

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

```
        int arr[]=new int[n];
       System.out.println("Enter the array");
       for(int i=0;i<n;i++)
       {
          arr[i]=sc.nextInt();
       }

       for(int i=0;i<n;i++)
       {
          for(int j=i+1;j<n;j++)
          {
             if(arr[i]<arr[j])
             {
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
             }
          }
       }

       System.out.println("Third Largest element is "+arr[2]);
    }
}
```

```
Enter the size of the array
5
Enter the array
22 11 44 55 33
Third Largest element is 33
```

17) Java Program to Find 2nd Largest Number in an array

```
import java.util.Scanner;
public class FindElement
{

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

        int arr[]=new int[n];
       System.out.println("Enter the array");
```

```
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }

        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                if(arr[i]<arr[j])
                {
                    int temp=arr[i];
                    arr[i]=arr[j];
                    arr[j]=temp;
                }
            }
        }

        System.out.println("Second Largest element is "+arr[1]);
    }
}
```

```
Enter the size of the array
5
Enter the array
2 3 5 0 7
Second Largest element is 5
```

18) Java Program to Find Largest Number in an array

```
import java.util.Scanner;

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

        int arr[]=new int[n];  /
```

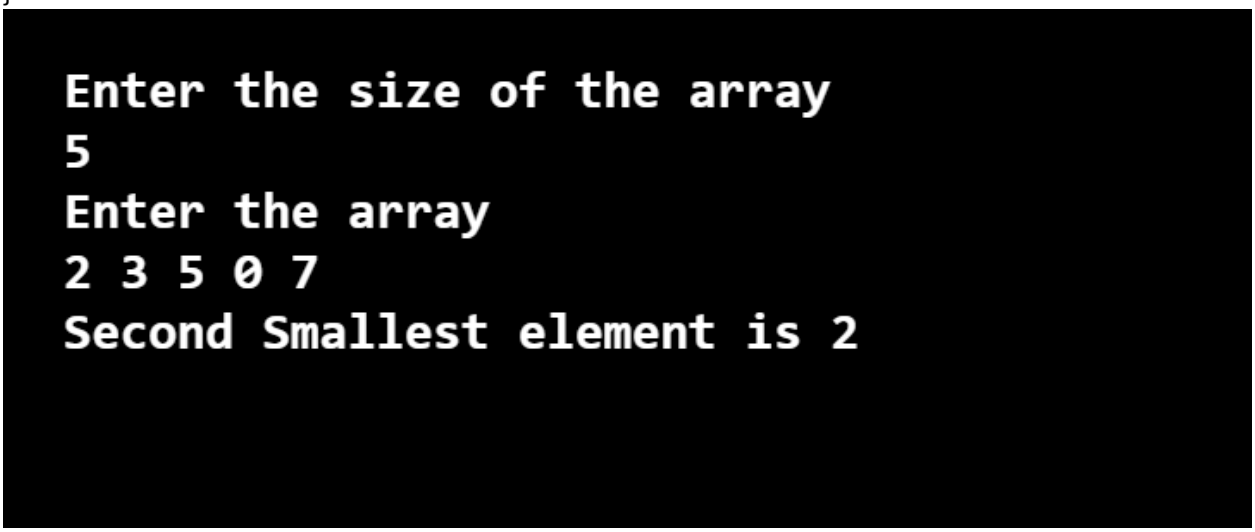
```
        System.out.println("Enter the array");
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }

        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                if(arr[i]<arr[j])
                {
                    int temp=arr[i];
                    arr[i]=arr[j];
                    arr[j]=temp;
                }
            }
        }

        System.out.println("Largest element is "+arr[0]);

    }
}
```

```
Enter the size of the array
5
Enter the array
0034 344 3554 376 030
Largest element is 3554
```

19) Java to Program Find 2nd Smallest Number in an array

```
import java.util.Scanner;
public class FindElement
{
   public static void main(String []args)
    {
        Scanner sc=new Scanner(System.in);
        int n;
        System.out.println("Enter the size of the array");
        n=sc.nextInt();

        int arr[]=new int[n]; //Declare array
       System.out.println("Enter the array");
       for(int i=0;i<n;i++)
       {
           arr[i]=sc.nextInt();
```

```
        }

        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                if(arr[i]<arr[j])
                {
                    int temp=arr[i];
                    arr[i]=arr[j];
                    arr[j]=temp;
                }
            }
        }

        System.out.println("Second Smallest element is "+arr[n-2]);
    }
}
```



20) Java Program to Find Smallest Number in an array

```
import java.util.Scanner;
public class Smallest
{
   public static void main(String[] args)
   {
      int t, i, small;
      Scanner scan = new Scanner(System.in);

      System.out.print("Enter the Size of Array: ");
      t = scan.nextInt();
      int[] arr = new int[tot];
      System.out.print("Enter " +t+ " Numbers: ");
```

```
      for(i=0; i<t; i++)
        arr[i] = scan.nextInt();

      small = arr[0];
      for(i=1; i<t; i++)
      {
        if(small>arr[i])
          small = arr[i];
      }

      System.out.println("\nSmallest Number = " +small);
    }
}
```

```
Enter the Size of Array: 5
Enter 5 Numbers: 33 44 08 097 002

Smallest Number = 2
```

21) Java Program to Remove Duplicate Element in an array

```
import java.util.Scanner;

public class RemoveDuplicateElementFromArray {
  public static void main(String[] args) {
    int[] arr_elements = new int[20];

    int initial_element, next_element;
    int i;

    Scanner sc = new Scanner(System.in);

    System.out.print("Enter array size: ");
    int arr_size = sc.nextInt();

    System.out.println("Read Array Elements From User :");

    for (i = 0; i < arr_size; ++i) {
      System.out.print("Enter array elements of index " + i + ": ");
      arr_elements[i] = sc.nextInt();
    }
```

```
    System.out.println("Before removing duplicate element array are :");
    for (i = 0; i < arr_size; ++i) {
      System.out.println(arr_elements[i]);
    }

    System.out.println();
    System.out.println("After removing duplicate element array are :");
    for (initial_element = 0; initial_element < arr_size; ++initial_element) {
    for (next_element = initial_element + 1; next_element < arr_size;) {
        /* if initial_element matches to next_element
        then take next _element and matches till end */
        if (arr_elements[initial_element] == arr_elements[next_element]) {
          for (int temp = next_element; temp < arr_size; ++temp) {
            arr_elements[temp] = arr_elements[temp + 1];
          }
          arr_size = arr_size - 1;
        } else
          next_element++;
      }
    }

    for (i = 0; i < arr_size; ++i)
      System.out.println(arr_elements[i]);
  }
}
```

```
Enter array size: 7
Read Array Elements From User :
Enter array elements of index 0: 3
Enter array elements of index 1: 1
Enter array elements of index 2: 2
Enter array elements of index 3: 3
Enter array elements of index 4: 3
Enter array elements of index 5: 2
Enter array elements of index 6: 1
Before removing duplicate element array are :
3
1
2
3
3
2
1
After removing duplicate element array are :
3
1
2
```

22) Java Program to Print Odd and Even Numbers from an array

```
import java.io.*;
import java.util.Scanner;

public class Main
{
   public static void main(String[] args)
   {
       Scanner sc=new Scanner(System.in);

      int n;    //Declare size of an array
      System.out.println("Enter the size of the array: ");
      n=sc.nextInt();
      int arr[]=new int[n];
      System.out.println("Enter the array elements: ");
      for(int i=0;i<n;i++)
      {
         arr[i]=sc.nextInt();
      }

      System.out.println("The Even Elements are...");
      for(int i=0;i<n;i++)
```

```
        {
            if(arr[i]%2==0)  //Check whether even or not
            {
                System.out.print(arr[i]+" ");
            }
        }
        System.out.println(" ");


        System.out.println("The Odd Elements are...");
        for(int i=0;i<n;i++)
        {
            if(arr[i]%2!=0)
            {
                System.out.print(arr[i]+" ");
            }
        }
    }
}
```

```
Enter the size of the array:
8
Enter the array elements:
2 3 5 0 7 1 4 8
The Even Elements are...
2 0 4 8
The Odd Elements are...
3 5 7 1
```

23) How to Sort an Array in Java

```
import java.util.Arrays;
import java.util.Scanner;
import java.util.Collections;

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

```
    Scanner sc=new Scanner(System.in);
    int n;
    System.out.println("Enter the number of elements :");
    n=sc.nextInt();

    Integer arr[]=new Integer[n];
    System.out.println("Enter the elements of the array :");
    for(int i=0;i<n;i++)
    {
        arr[i]=sc.nextInt();
    }

    int temp = 0;

     for (int i = 0; i < arr.length; i++)
     {
        for (int j = i+1; j < arr.length; j++)
        {
           if(arr[i] > arr[j])
           {
              temp = arr[i];
              arr[i] = arr[j];
              arr[j] = temp;
           }
        }
    }

    System.out.println();

    System.out.println("Elements of array sorted in ascending order: ");
    for (int i = 0; i < arr.length; i++)
    {
        System.out.print(arr[i] + " ");
    }
}
```

```
Enter the number of elements :
9
Enter the elements of the array :
2 3 5 0 7 9 1 8 4

Elements of array sorted in ascending order:
0 1 2 3 4 5 7 8 9
```

Java Matrix Programs

1) Java Programs to create matrix of any valid order

```
import java.util.Scanner;
public class MatrixUserInput
{
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("Please enter number of matrix rows : ");
      int row = sc.nextInt();
      System.out.println("Please enter number of matrix columns : ");
      int col = sc.nextInt();
      // defining two dimensional array java
      int[][] numbers = new int[row][col];
      // filling java matrix
      fillingMatrix(sc, numbers, row, col);
      // printing 2d array in matrix form in java
      printingMatrix(numbers, row, col);
   }
   public static void fillingMatrix(Scanner scan, int num[][], int rows, int columns)
   {
      System.out.println("Please enter elements in matrix : ");
      for(int a = 0; a < rows; a++)
      {
         for(int b = 0; b < columns; b++)
         {
            num[a][b] = scan.nextInt();
         }
      }
   }
   public static void printingMatrix(int num[][], int rows, int columns)
   {
      System.out.println("Printing 2d array in matrix form : ");
      for(int a = 0; a < rows; a++)
      {
         for(int b = 0; b < columns; b++)
         {
            System.out.print(num[a][b] + "\t");
         }
         System.out.println();
      }
   }
}
```

```
Please enter number of matrix rows :
3
Please enter number of matrix columns :
3
Please enter elements in matrix :
1 2 3 4 5 6 7 8 9
Printing 2d array in matrix form :
1   2   3
4   5   6
7   8   9
```

2) Java Program to Add Two Matrices

```
import java.util.Scanner;
public class Main
{
   public static void main(String[] args)
   {
      int p, q, m, n;    //Declare matrix size
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number of rows in the first matrix:");
      p = sc.nextInt();
      System.out.print("Enter the number of columns in the first matrix:");
      q = sc.nextInt();
      System.out.print("Enter the number of rows in the second matrix:");
      m = sc.nextInt();
      System.out.print("Enter the number of columns in the second matrix:");
      n = sc.nextInt();          if (p == m && q == n)
      {
         int a[][] = new int[p][q];
         int b[][] = new int[m][n];
         int c[][] = new int[m][n];
         //Initialize the first matrix
         System.out.println("Enter all the elements of first matrix:");
         for (int i = 0; i < p; i++)
         {
            for (int j = 0; j < q; j++)
            {
               a[i][j] = sc.nextInt();
            }
         }
         System.out.println("");
         //Initialize the second matrix
```

```
System.out.println("Enter all the elements of second matrix:");
for (int i = 0; i < m; i++)
{
   for (int j = 0; j < n; j++)
   {
      b[i][j] = sc.nextInt();
   }
}
System.out.println("");
System.out.println("First Matrix:");
for (int i = 0; i < p; i++)
{
   for (int j = 0; j < q; j++)
   {
      System.out.print(a[i][j]+" ");
   }
   System.out.println("");
}

System.out.println("Second Matrix:");
for (int i = 0; i < m; i++)
{
   for (int j = 0; j < n; j++)
   {
      System.out.print(b[i][j]+" ");
   }
   System.out.println("");
}

for (int i = 0; i < p; i++)
{
   for (int j = 0; j < n; j++)
   {
      for (int k = 0; k < q; k++)
      {
         c[i][j] = a[i][j] + b[i][j];
      }
   }
}


System.out.println("Matrix after addition:");
for (int i = 0; i < p; i++)
{
   for (int j = 0; j < n; j++)
   {
      System.out.print(c[i][j]+" ");
   }
```

```
            System.out.println("");
          }
        }
        else
        {
          System.out.println("Addition not possible");
          System.out.println("Try Again");
        }
    }
}
```

```
Enter the number of rows in the first matrix:2
Enter the number of columns in the first matrix:2
Enter the number of rows in the second matrix:2
Enter the number of columns in the second matrix:2
Enter all the elements of first matrix:
1 2 3 4

Enter all the elements of second matrix:
4 3 2 1

First Matrix:
1 2
3 4
Second Matrix:
4 3
2 1
Matrix after addition:
5 5
5 5
```

3) Java Program to Multiply Two Matrices

```
import java.util.Scanner;

public class MatrixMultiplication
{
  public static void main(String args[])
  {
    int m, n, p, q, sum = 0, c, d, k;

    Scanner in = new Scanner(System.in);
    System.out.println("Enter the number of rows and columns of first matrix");
    m = in.nextInt();
```

```
n = in.nextInt();

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

System.out.println("Enter elements of first matrix");

for (c = 0; c < m; c++)
  for (d = 0; d < n; d++)
    first[c][d] = in.nextInt();

System.out.println("Enter the number of rows and columns of second matrix");
p = in.nextInt();
q = in.nextInt();

if (n != p)
  System.out.println("The matrices can't be multiplied with each other.");
else
{
  int second[][] = new int[p][q];
  int multiply[][] = new int[m][q];

  System.out.println("Enter elements of second matrix");

  for (c = 0; c < p; c++)
    for (d = 0; d < q; d++)
      second[c][d] = in.nextInt();

  for (c = 0; c < m; c++)
  {
    for (d = 0; d < q; d++)
    {
      for (k = 0; k < p; k++)
      {
        sum = sum + first[c][k]*second[k][d];
      }

      multiply[c][d] = sum;
      sum = 0;
    }
  }

  System.out.println("Product of the matrices:");

  for (c = 0; c < m; c++)
  {
    for (d = 0; d < q; d++)
      System.out.print(multiply[c][d]+"\t");
```

```
            System.out.print("\n");
        }
    }
  }
}
```

```
Enter the number of rows and columns of first matrix
3 3
Enter elements of first matrix
1 2 3 1 2 3 1 1 0
Enter the number of rows and columns of second matrix
3 3
Enter elements of second matrix
1 1 0 1 1 1 2 1 0
Product of the matrices:
9   6   2
9   6   2
2   2   1
```

4) Java Program to subtract the two matrices

```
import java.util.Scanner;

public class CodesCracker
{
   public static void main(String[] args)
   {
      int i, j;
      Scanner scan = new Scanner(System.in);

      System.out.print("Enter Row Size of First Matrix: ");
      int rOne = scan.nextInt();
      System.out.print("Enter Column Size of First Matrix: ");
      int cOne = scan.nextInt();
      int[][] mat1 = new int[rOne][cOne];
      System.out.print("Enter " +(rOne*cOne)+ " Elements for First Matrix: ");
      for(i=0; i<rOne; i++)
      {
         for(j=0; j<cOne; j++)
            mat1[i][j] = scan.nextInt();
      }
      System.out.print("Enter Row Size of Second Matrix: ");
      int rTwo = scan.nextInt();
      System.out.print("Enter Column Size of Second Matrix: ");
      int cTwo = scan.nextInt();
      int[][] mat2 = new int[rTwo][cTwo];
      System.out.print("Enter " +(rTwo*cTwo)+ " Elements for Second Matrix: ");
      for(i=0; i<rTwo; i++)
```

```
    {
      for(j=0; j<cTwo; j++)
        mat2[i][j] = scan.nextInt();
    }

    if(rOne==rTwo && cOne==cTwo)
    {
      int[][] mat3 = new int[rOne][cOne];
      for(i=0; i<rOne; i++)
      {
        for(j=0; j<cOne; j++)
          mat3[i][j] = mat1[i][j] - mat2[i][j];
      }

      System.out.println("\nResult of Matrix 1 - Matrix 2 is:");
      for(i=0; i<rOne; i++)
      {
        for(j=0; j<cOne; j++)
          System.out.print(mat3[i][j]+ "  ");
        System.out.print("\n");
      }
    }
    else
      System.out.println("\nOrder Mismatched!");
  }
}
```

```
Enter Row Size of First Matrix: 3
Enter Column Size of First Matrix: 3
Enter 9 Elements for First Matrix: 9 8 7 6 5 4 3 2 1
Enter Row Size of Second Matrix: 3
Enter Column Size of Second Matrix: 3
Enter 9 Elements for Second Matrix: 8 7 6 5 4 3 2 1 1

Result of Matrix 1 - Matrix 2 is:
1  1  1
1  1  1
1  1  0
```

```
Enter Row Size of First Matrix: 2
Enter Column Size of First Matrix: 2
Enter 4 Elements for First Matrix: 1 2 3 4
Enter Row Size of Second Matrix: 3
Enter Column Size of Second Matrix: 2
Enter 6 Elements for Second Matrix: 1 2 3 4 5 6

Order Mismatched!
```

5) Java Program to determine whether two matrices are equal

```
import java.util.Scanner;

public class MatrixEqualOrNot {
	private static Scanner sc;

	public static void main(String[] args) {
		int i, j, rows, columns, isEqual = 1;

		sc= new Scanner(System.in);

		System.out.println("\n Enter Matrix Rows & Columns :  ");
		rows = sc.nextInt();
		columns = sc.nextInt();

		int[][] x = new int[rows][columns];
		int[][] y = new int[rows][columns];

		System.out.println("\n Enter the First Matrix Items : ");
		for(i = 0; i < rows; i++) {
			for(j = 0; j < columns; j++) {
				x[i][j] = sc.nextInt();
			}
		}

		System.out.println("\n Enter the Second Matrix Items : ");
		for(i = 0; i < rows; i++) {
			for(j = 0; j < columns; j++) {
				y[i][j] = sc.nextInt();
```

```
            }
        }

        for(i = 0; i < x.length; i++)
        {
            for(j = 0; j < x[0].length; j++)
            {
                if(x[i][j] != y[i][j]) {
                    isEqual = 0;
                    break;
                }
            }
        }
        if(isEqual == 1) {
            System.out.println("\nMatrix x is Equal to Matrix y");
        }
        else {
            System.out.println("\nMatrix x is Not Equal to Matrix y");
        }
```

```
 Enter Matrix Rows & Columns :
2 2

 Enter the First Matrix Items :
 1 2
 3 4

 Enter the Second Matrix Items :
 1 2
 3 4

Matrix x is Equal to Matrix y
```

```
 Enter Matrix Rows & Columns :
2 2

 Enter the First Matrix Items :
1 2
3 4

 Enter the Second Matrix Items :
2 1
3 4

Matrix x is Not Equal to Matrix y
```

6) Java Program to display the lower triangular matrix

```
import java.util.*;

public class Main
{
    // Print the matrix
    public static void printMatrix(int[][] arr)
    {
        int m = arr.length;  //For Rows
        int n = arr[0].length; //For columns
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }

    //Display the lower triangular matrix
    public static void lowerTriangularMatrix(int arr[][])
    {
```

```
    int m = arr.length;
    int n = arr[0].length;

    if (m != n)
    {
        System.out.println("Matrix entered should be a Square Matrix");
        System.out.println("Try Again..");
        return;
    }
    else
    {
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if (i < j)
                {
                    arr[i][j] = 0;
                }
            }
        }

        System.out.println( "Lower Triangular Matrix is : ");

        printMatrix(arr);
    }
}
public static void main(String[] args)
{
    Scanner sc=new Scanner(System.in);

    int m,n;
    System.out.println("Enter the number of rows: ");
    m=sc.nextInt();

    System.out.println("Enter the number of columns: ");
    n=sc.nextInt();

    System.out.println("Enter the Matrix Elements: ");
    int arr[][] = new int[m][n];
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            arr[i][j]=sc.nextInt();
        }
    }
    System.out.println( "Original Matrix is : ");
```

```
        printMatrix(arr);

        lowerTriangularMatrix(arr);
    }
}
```

```
Enter the number of rows:
3
Enter the number of columns:
3
Enter the Matrix Elements:
1 2 3 4 5 6 7 8 9
Original Matrix is :
1 2 3
4 5 6
7 8 9
Lower Triangular Matrix is :
1 0 0
4 5 0
7 8 9
```

7) Java Program to display the upper triangular matrix

```
import java.util.*;
public class Main
{
    public static void printMatrix(int[][] arr)
    {
        int m = arr.length;
        int n = arr[0].length;
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
    //Display the upper triangular matrix
```

```
public static void upperTriangularMatrix(int arr[][])
{
    int m = arr.length;
    int n = arr[0].length;
    if (m != n)
    {
        System.out.println("Matrix entered should be a Square Matrix");
        System.out.println("Try Again..");
        return;
    }
    else
    {
        // looping over the whole matrix
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if (i > j)
                {
                    arr[i][j] = 0;
                }
            }
        }
        System.out.println( "Upper Triangular Matrix is : ");
        printMatrix(arr);
    }
}
public static void main(String[] args)
{
    Scanner sc=new Scanner(System.in);
    int m,n;
    System.out.println("Enter the number of rows: ");
    m=sc.nextInt();
    System.out.println("Enter the number of columns: ");
    n=sc.nextInt();
    System.out.println("Enter the matrix elements: ");
    int arr[][] = new int[m][n];
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            arr[i][j]=sc.nextInt();
        }
    }

    System.out.println( "Original Matrix is : ");
    printMatrix(arr);
            upperTriangularMatrix(arr);
```

```
    }
}
```

```
Enter the number of rows:
3
Enter the number of columns:
3
Enter the matrix elements:
9 8 7 6 1 2 3 4 5
Original Matrix is :
9 8 7
6 1 2
3 4 5
Upper Triangular Matrix is :
9 8 7
0 1 2
0 0 5
```

8) Java Program to find the frequency of odd & even numbers in the given matrix

```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        int m,n;
         Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of rows in matrix:");
        m = sc.nextInt();

        System.out.print("Enter number of columns in matrix:");
        n = sc.nextInt();
```

```
        int arr[][] = new int[m][n];
        System.out.println("Enter all the elements of matrix:");
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                arr[i][j] = sc.nextInt();
            }
        }
                System.out.println("The Original Matrix:");
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println("");
        }

        int even=0,odd=0;

        //Use for loops to iterate through the matrix elements
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                if(arr[i][j]%2==0)
                {
                    even++;                    }
                else
                {
                    odd++;
                }
            }
        }
        System.out.println("Total Odd Number in the Matrix: " + odd);
        System.out.println("Total Even Number in the Matrix: " + even);

    }
}
```

```
Enter number of rows in matrix:3
Enter number of columns in matrix:3
Enter all the elements of matrix:
1 2 3 4  9 8 7 6 5
The Original Matrix:
1 2 3
4 9 8
7 6 5
Total Odd Number in the Matrix: 5
Total Even Number in the Matrix: 4
```

9) Java Program to find the product of two matrices

```
import java.util.Scanner;

public class MatrixMultiplication
{
  public static void main(String args[])
  {
    int m, n, p, q, sum = 0, c, d, k;

    Scanner in = new Scanner(System.in);
    System.out.println("Enter the number of rows and columns of first matrix");
    m = in.nextInt();
    n = in.nextInt();

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

    System.out.println("Enter elements of first matrix");

    for (c = 0; c < m; c++)
      for (d = 0; d < n; d++)
        first[c][d] = in.nextInt();

    System.out.println("Enter the number of rows and columns of second matrix");
    p = in.nextInt();
    q = in.nextInt();
```

```
    if (n != p)
      System.out.println("The matrices can't be multiplied with each other.");
    else
    {
      int second[][] = new int[p][q];
      int multiply[][] = new int[m][q];

      System.out.println("Enter elements of second matrix");

      for (c = 0; c < p; c++)
        for (d = 0; d < q; d++)
          second[c][d] = in.nextInt();

      for (c = 0; c < m; c++)
      {
        for (d = 0; d < q; d++)
        {
          for (k = 0; k < p; k++)
          {
            sum = sum + first[c][k]*second[k][d];
          }

          multiply[c][d] = sum;
          sum = 0;
        }
      }

      System.out.println("Product of the matrices:");

      for (c = 0; c < m; c++)
      {
        for (d = 0; d < q; d++)
          System.out.print(multiply[c][d]+"\t");

        System.out.print("\n");
      }
    }
  }
}
```

```
Enter the number of rows and columns of first matrix
3 3
Enter elements of first matrix
1 2 3 1 2 3 1 1 0
Enter the number of rows and columns of second matrix
3 3
Enter elements of second matrix
1 1 0 1 1 1 2 1 0
Product of the matrices:
9   6   2
9   6   2
2   2   1
```

10) Java Program to find the sum of each row and each column of a matrix

```
import java.util.Scanner;
public class Main
{
   public static void main(String[] args)
   {
      int p, q, m, n;    //Declare matrix size
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number of rows in the first matrix:");
      p = sc.nextInt();
      System.out.print("Enter the number of columns in the first matrix:");
      q = sc.nextInt();
      System.out.print("Enter the number of rows in the second matrix:");
      m = sc.nextInt();
      System.out.print("Enter the number of columns in the second matrix:");
      n = sc.nextInt();          if (p == m && q == n)
      {
         int a[][] = new int[p][q];
         int b[][] = new int[m][n];
         int c[][] = new int[m][n];
         //Initialize the first matrix
         System.out.println("Enter all the elements of first matrix:");
         for (int i = 0; i < p; i++)
         {
            for (int j = 0; j < q; j++)
            {
               a[i][j] = sc.nextInt();
            }
         }
         System.out.println("");
         //Initialize the second matrix
         System.out.println("Enter all the elements of second matrix:");
         for (int i = 0; i < m; i++)
```

```
{
   for (int j = 0; j < n; j++)
   {
      b[i][j] = sc.nextInt();
   }
}
System.out.println("");
System.out.println("First Matrix:");
for (int i = 0; i < p; i++)
{
   for (int j = 0; j < q; j++)
   {
      System.out.print(a[i][j]+" ");
   }
   System.out.println("");
}

System.out.println("Second Matrix:");
for (int i = 0; i < m; i++)
{
   for (int j = 0; j < n; j++)
   {
      System.out.print(b[i][j]+" ");
   }
   System.out.println("");
}

for (int i = 0; i < p; i++)
{
   for (int j = 0; j < n; j++)
   {
      for (int k = 0; k < q; k++)
      {
         c[i][j] = a[i][j] + b[i][j];
      }
   }
}


System.out.println("Matrix after addition:");
for (int i = 0; i < p; i++)
{
   for (int j = 0; j < n; j++)
   {
      System.out.print(c[i][j]+" ");
   }
   System.out.println("");
}
```

```
        }
        else
        {
            System.out.println("Addition not possible");
            System.out.println("Try Again");
        }
    }
}
```

```
Enter the number of rows in the first matrix:2
Enter the number of columns in the first matrix:2
Enter the number of rows in the second matrix:2
Enter the number of columns in the second matrix:2
Enter all the elements of first matrix:
1 2 3 4

Enter all the elements of second matrix:
4 3 2 1

First Matrix:
1 2
3 4
Second Matrix:
4 3
2 1
Matrix after addition:
5 5
5 5
```

11) Java Program to find the transpose of a given matrix

```
import java.util.*;

public class Main
{
    public static void main(String []args)
    {
        ///Take input from the user
        Scanner sc=new Scanner(System.in);

        int m,n;                //Matrix Size Declaration

        System.out.println("Enter the number of rows: \n");
        m=sc.nextInt();  //Matrix Size Initialization
```

```
        System.out.println("Enter the number of column: \n");
        n=sc.nextInt();  //Matrix Size Initialization

        int arr[][]=new int[10][10]; //Matrix Size Declaration
        System.out.println("Enter the elements of the matrix: ");
        for(int i=0;i<m;i++)  //Matrix Initialization
        {
          for(int j=0;j<n;j++)
          {
              arr[i][j]=sc.nextInt();
          }
        }

        //Print the original Matrix
        System.out.println("The elements in the original matrix are: ");
        for(int i=0;i<m;i++)     //Print the matrix
        {
           for(int j=0;j<n;j++)
           {
               System.out.print(arr[i][j]+" ");
           }
          System.out.println("");
       }

       int brr[][]=new int[10][10];        //Transpose Matrix Declaration
       for(int i=0;i<m;i++)     //Transpose Matrix initialization
       {
           for(int j=0;j<n;j++)
           {
              brr[j][i]=arr[i][j];     //Store elements in the transpose matrix
           }
       }

       System.out.println("After transposing the elements are...");
       for(int i=0;i<m;i++)      //Print the transpose matrix
       {
           for(int j=0;j<n;j++)
           {
              System.out.print(brr[i][j]+" ");
           }
          System.out.println("");
       }

    }
}
```

```
Enter the number of rows:

3
Enter the number of column:

3
Enter the elements of the matrix:
2 3 5 0 7 1 2 3 4
The elements in the original matrix are:
2 3 5
0 7 1
2 3 4
After transposing the elements are...
2 0 2
3 7 3
5 1 4
```

12) Java Program to determine whether a given matrix is an identity matrix

```
import java.util.Scanner;

public class IdentityMatrix {
	private static Scanner sc;

	public static void main(String[] args) {

		int i, j, rows, columns, Flag = 1;

		sc= new Scanner(System.in);

		System.out.println("\n Please Enter Identity Matrix Rows and Columns : ");
		rows = sc.nextInt();
		columns = sc.nextInt();

		int[][] arr = new int[rows][columns];

		System.out.println("\n Please Enter the Identity Matrix Items : ");
		for(i = 0; i < rows; i++) {
			for(j = 0; j < columns; j++) {
```

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

        for(i = 0; i < rows ; i++)
        {
                for(j = 0; j < columns; j++)
                {
                        if(arr[i][j] != 1 && arr[j][i] != 0) {
                                Flag = 0;
                                break;
                        }
                }
        }
        if(Flag == 1) {
                System.out.println("\nMatrix is an Identity Matrix");
        }
        else {
                System.out.println("\nMatrix is Not an Identity Matrix");
        }
    }
}
```

```
 Please Enter Identity Matrix Rows and Columns :
 3 3

 Please Enter the Identity Matrix Items :
1 0 0
0 1 0
0 0 1

Matrix is an Identity Matrix
```

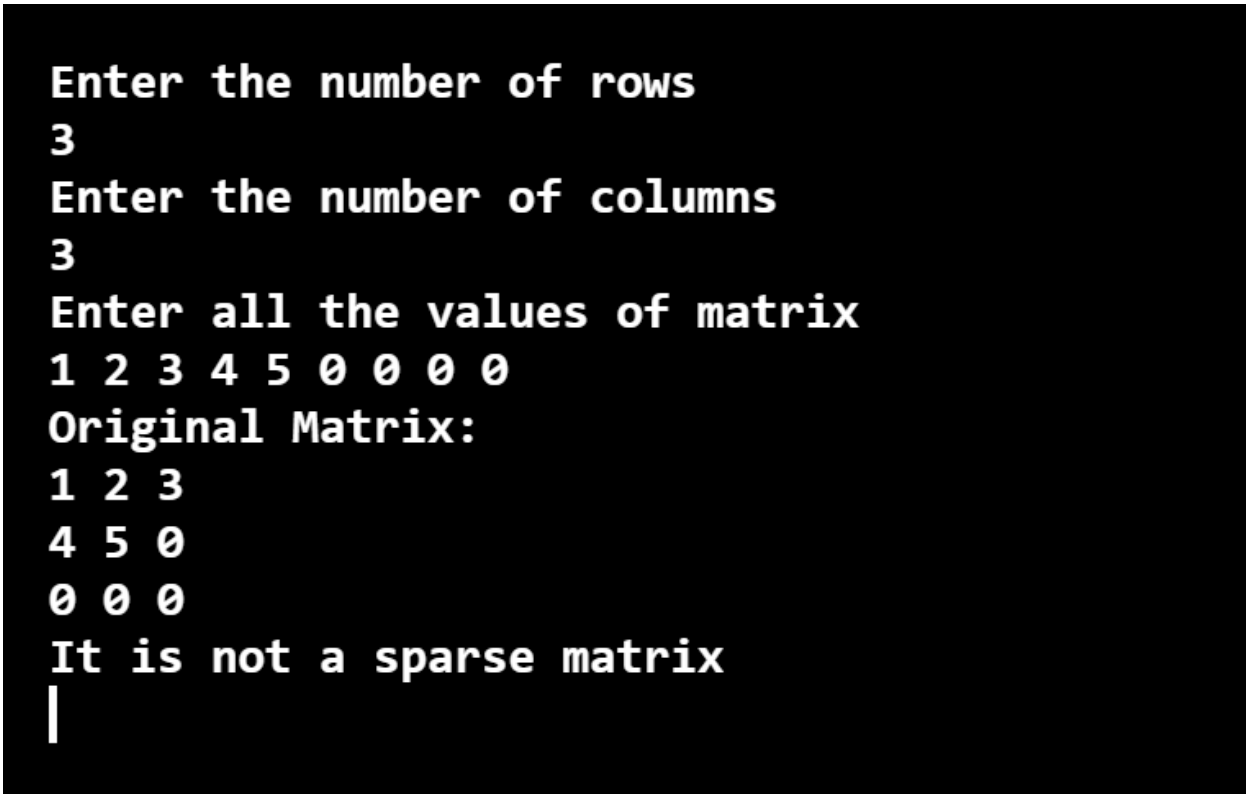
```
 Please Enter Identity Matrix Rows and Columns :
3 3

 Please Enter the Identity Matrix Items :
2 0 0
0 2 0
0 0 2

Matrix is Not an Identity Matrix
```

13) Java Program to determine whether a given matrix is a sparse matrix

```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        // declare variables
        int m, n;
        // To take input from the user
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of rows ");
        // Initialize the number of rows
        m = sc.nextInt();
        System.out.println("Enter the number of columns ");
        // Initialize the number of columns
        n = sc.nextInt();
        // declare a mxn order array
        int a[][] = new int[m][n];
        System.out.println("Enter all the values of matrix ");
        // Initialize the matrix elements
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                a[i][j] = sc.nextInt();
            }
        }
        System.out.println("Original Matrix:");
        // print the original matrix
```

```
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                System.out.print(a[i][j] + " ");
            }
            System.out.println("");
        }
        int size= m*n;  //Stores the size of the matrix
        int count=0;    //Variable to check for the number of 0 elements
        //Loop to count all zero element present in matrix
        for(int i = 0; i < m; i++)
        {
            for(int j = 0; j < n; j++)
            {
                if(a[i][j] == 0)    //Check if element is 0 or not
                    count++;    //Increment the count if 0 element is found
            }
        }
        if(count>(size/2))
        System.out.println("It is a sparse matrix");
        else
        System.out.println("It is not a sparse matrix");
    }
}
```

```
Enter the number of rows
3
Enter the number of columns
3
Enter all the values of matrix
2 3 0 0 0 0 0 0 0
Original Matrix:
2 3 0
0 0 0
0 0 0
It is a sparse matrix
```



14) Java Program to transpose matrix

```
import java.util.*;

public class Main
{
    public static void main(String []args)
    {
       ///Take input from the user
       Scanner sc=new Scanner(System.in);

       int m,n;                //Matrix Size Declaration

       System.out.println("Enter the number of rows: \n");
       m=sc.nextInt();  //Matrix Size Initialization

       System.out.println("Enter the number of column: \n");
       n=sc.nextInt();  //Matrix Size Initialization

       int arr[][]=new int[10][10]; //Matrix Size Declaration
       System.out.println("Enter the elements of the matrix: ");
       for(int i=0;i<m;i++)  //Matrix Initialization
       {
```

```
            for(int j=0;j<n;j++)
            {
                arr[i][j]=sc.nextInt();
            }
        }

        //Print the original Matrix
        System.out.println("The elements in the original matrix are: ");
        for(int i=0;i<m;i++)     //Print the matrix
        {
            for(int j=0;j<n;j++)
            {
                System.out.print(arr[i][j]+" ");
            }
            System.out.println("");
        }

        int brr[][]=new int[10][10];        //Transpose Matrix Declaration
        for(int i=0;i<m;i++)     //Transpose Matrix initialization
        {
            for(int j=0;j<n;j++)
            {
                brr[j][i]=arr[i][j];     //Store elements in the transpose matrix
            }
        }

        System.out.println("After transposing the elements are...");
        for(int i=0;i<m;i++)      //Print the transpose matrix
        {
            for(int j=0;j<n;j++)
            {
                System.out.print(brr[i][j]+" ");
            }
            System.out.println("");
        }

    }
}
```

```
Enter the number of rows:

3
Enter the number of column:

3
Enter the elements of the matrix:
2 3 5 0 7 1 2 3 4
The elements in the original matrix are:
2 3 5
0 7 1
2 3 4
After transposing the elements are...
2 0 2
3 7 3
5 1 4
```

Java Searching and Sorting Programs

1) Linear Search in Java

```
import java.util.Scanner;

public class LinearSearch
{
   public static void main(String[] args)
   {
      int i, pos=0;
      Scanner s = new Scanner(System.in);

      System.out.print("Enter the Size of Array: ");
```

```
    int n = s.nextInt();
    int[] arr = new int[n];
    System.out.print("Enter " +n+" Elements: ");
    for(i=0; i<n; i++)
      arr[i] = s.nextInt();

    System.out.print("Enter an Element to Search: ");
    int num = s.nextInt();

    for(i=0; i<n; i++)
    {
      if(num==arr[i])
      {
        pos = i+1;
        break;
      }
    }
    if(pos==0)
      System.out.println("\nThe element not found!");
    else
      System.out.println("\nThe element found at position: " +pos);
  }
}
```

```
Enter the Size of Array: 5
Enter 5 Elements: 2 3 5 0 7
Enter an Element to Search: 5

The element found at position: 3
```

```
Enter the Size of Array: 5
Enter 5 Elements: 2 3 5 0 7
Enter an Element to Search: 6

The element not found!
```

2) Binary Search in Java

```
import java.util.Scanner;
public class BinarySearchExample
{
   public static void main(String args[])
   {
      int counter, num, item, array[], first, last, middle;
      //To capture user input
      Scanner input = new Scanner(System.in);
      System.out.println("Enter number of elements:");
      num = input.nextInt();

      //Creating array to store the all the numbers
      array = new int[num];

      System.out.println("Enter " + num + " integers");
      //Loop to store each numbers in array
      for (counter = 0; counter < num; counter++)
          array[counter] = input.nextInt();

      System.out.println("Enter the search value:");
      item = input.nextInt();
      first = 0;
      last = num - 1;
      middle = (first + last)/2;

      while( first <= last )
      {
        if ( array[middle] < item )
          first = middle + 1;
        else if ( array[middle] == item )
        {
          System.out.println(item + " found at location " + (middle + 1) + ".");
          break;
        }
        else
        {
            last = middle - 1;
        }
        middle = (first + last)/2;
      }
      if ( first > last )
          System.out.println(item + " is not found.\n");
   }
}
```

```
Enter number of elements:
5
Enter 5 integers
11 22 33 44 55
Enter the search value:
33
33 found at location 3.
```

```
Enter number of elements:
4
Enter 4 integers
11 22 33 44
Enter the search value:
55
55 is not found.
```

3) Bubble Sort in Java

```
import java.util.Scanner;

public class BubbleSort
{
   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
```

```
    System.out.print("Enter the Size: ");
    int n = s.nextInt();

    int[] arr = new int[n];

    System.out.print("Enter " +n+" Elements in Random Order: ");
    for(int i=0; i<n; i++)
      arr[i] = s.nextInt();

    System.out.println("\n\nSorting the array...");
    for(int i=0; i<(n-1); i++)
    {
      for(int j=0; j<(n-i-1); j++)
      {
        if(arr[j]>arr[j+1])
        {
          int temp = arr[j];
          arr[j] = arr[j+1];
          arr[j+1] = temp;
        }
      }
    }
    System.out.println("The array is sorted successfully!");

    System.out.println("\nThe new sorted array is:");
    for(int i=0; i<n; i++)
      System.out.print(arr[i]+ " ");
  }
}
```

```
Enter the Size: 5
Enter 5 Elements in Random Order: 2 3 5 0 7


Sorting the array...
The array is sorted successfully!

The new sorted array is:
0 2 3 5 7
```

4) Selection Sort in Java

```
import java.util.Scanner;

public class SelectionSort
{
   public static void main(String[] args)
   {
      int tot, i, j, count, small, index=0, x;
      Scanner scan = new Scanner(System.in);

      System.out.print("Enter the Size of Array: ");
      tot = scan.nextInt();
      int[] arr = new int[tot];

      System.out.print("Enter " +tot+ " Elements for the Array: ");
      for(i=0; i<tot; i++)
         arr[i] = scan.nextInt();

      for(i=0; i<(tot-1); i++)
      {
         count=0;
         small = arr[i];
         for(j=(i+1); j<tot; j++)
         {
            if(small>arr[j])
            {
               small = arr[j];
               count++;
               index = j;
            }
         }
         if(count!=0)
         {
            x =  arr[i];
            arr[i] = small;
            arr[index] = x;
         }
      }

      System.out.println("\nThe new sorted array is: ");
      for(i=0; i<tot; i++)
         System.out.print(arr[i]+ " ");

   }
}
```

```
Enter the Size of Array: 5
Enter 5 Elements for the Array: 2 3 5 0 7

The new sorted array is:
0 2 3 5 7
```

5) Insertion Sort in Java

```
import java.util.Scanner;

public class InsertionSort
{
   public static void main(String[] args)
   {
      int n, i, j, element;
      Scanner scan = new Scanner(System.in);

      System.out.print("Enter the Size of Array: ");
      n = scan.nextInt();
      int[] arr = new int[n];
      System.out.print("Enter " +n+ " Elements: ");
      for(i=0; i<n; i++)
         arr[i] = scan.nextInt();

      for(i=1; i<n; i++)
      {
         element = arr[i];

         for(j=(i-1); j>=0 && arr[j]>element; j--)
            arr[j+1] = arr[j];

         arr[j+1] = element;
      }

      System.out.println("\nThe new sorted array is: ");
      for(i=0; i<n; i++)
         System.out.print(arr[i]+ " ");
   }
}
```

```
Enter the Size of Array: 6
Enter 6 Elements: 55 44 11 99 764 2

The new sorted array is:
2 11 44 55 99 764
```

Java Conversion Programs

1) How to convert String to int in Java

```
public class StringToInt{
public static void main(String args[]){
String s="200";
int i=Integer.parseInt(s);
System.out.println(i);
}}
```

```
200
```

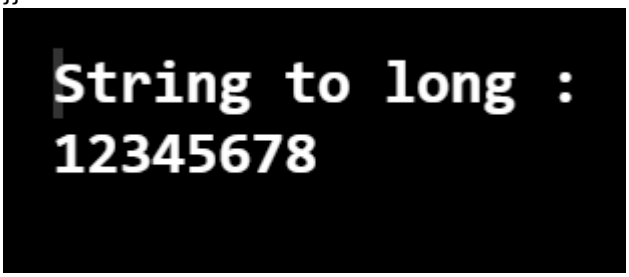
2) How to convert int to String in Java

```
public class IntToString{
public static void main(String args[]){
int i=200;
String  s=String.valueOf(i);
System.out.println(i+100);
System.out.println(s+100);
}}
```

```
300
200100
```

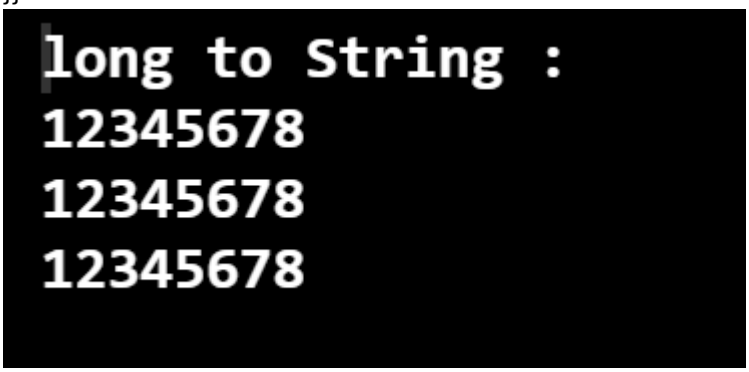
3) How to convert String to long in Java

```
public class StringToLong{
public static void main(String args[]){
System.out.println("String to long :");
        long l1=Long.parseLong("12345678");
System.out.println(l1);
}}
```



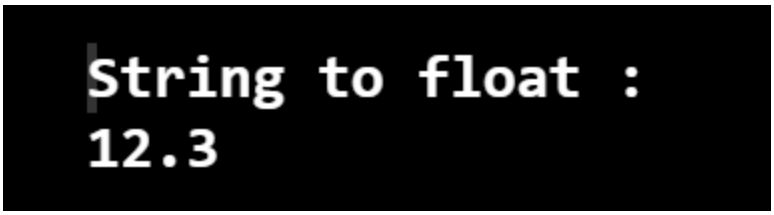
4) How to convert long to String in Java

```
public class LongToString{
public static void main(String args[]){
System.out.println("long to String :");
        String s7=Long.toString(12345678L);
System.out.println(s7);
        s7=String.valueOf(12345678L);
System.out.println(s7);
        s7=12345678L+"";
System.out.println(s7);
}}
```



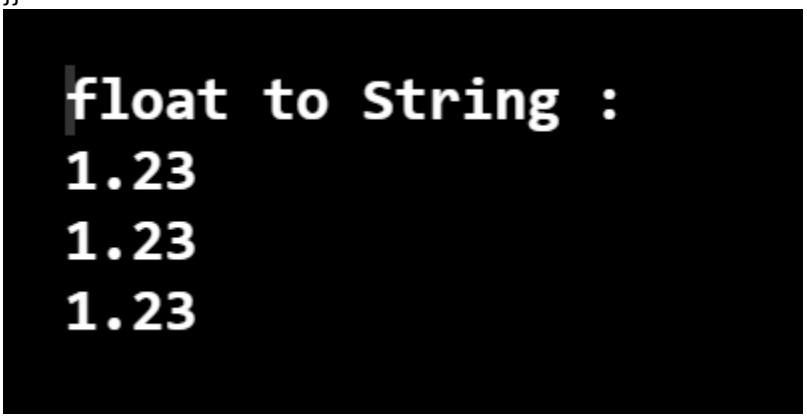
5) How to convert String to float in Java

```
public class StringtoFloat{
public static void main(String args[]){
System.out.println("String to float :");
        float f1=Float.parseFloat("12.3");
System.out.println(f1);
}}
```



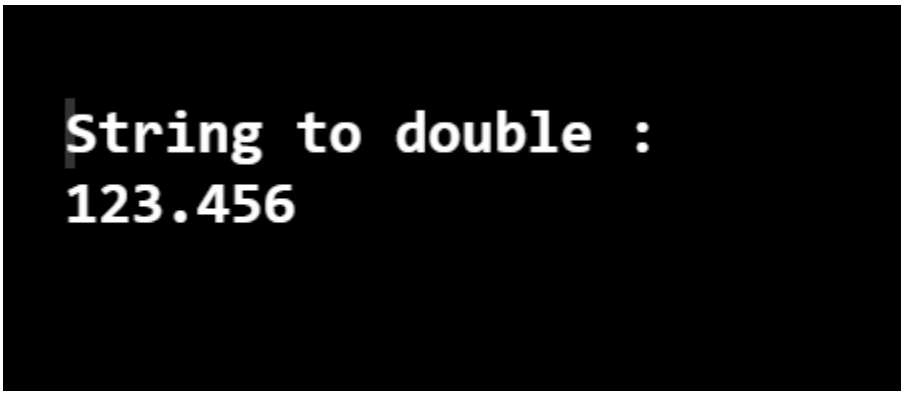
6) How to convert float to String in Java

```
public class FloattoString{
public static void main(String args[]){
System.out.println("float to String :");
        String s8=Float.toString(1.23F);
System.out.println(s8);
        s8=String.valueOf(1.23F);
System.out.println(s8);
        s8=1.23F+"";
System.out.println(s8);
}}
```



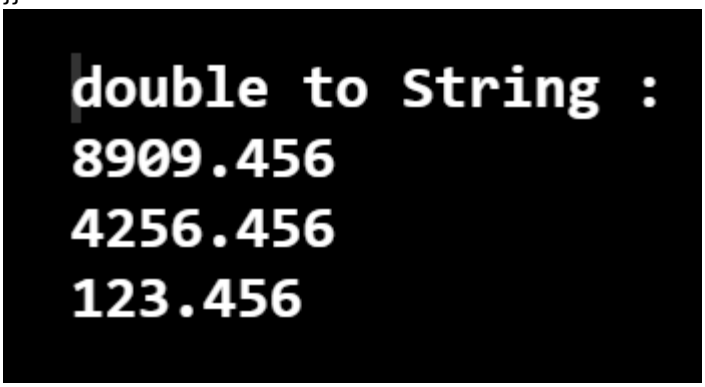
7) How to convert String to double in Java

```
public class StringtoDouble{
public static void main(String args[]){
System.out.println("String to double :");
      double d1=Double.parseDouble("123.456");
System.out.println(d1);
}}
```



8) How to convert double to String in Java

```
public class DoubleToString{
public static void main(String args[]){
System.out.println("double to String :");
        String s9=Double.toString(8909.456);
System.out.println(s9);
        s9=String.valueOf(4256.456);
System.out.println(s9);
        s9=123.456+"";
System.out.println(s9);
}}
```



9) How to convert String to Date in Java

```
import java.text.SimpleDateFormat;
import java.util.Date;
public class StringToDate {
public static void main(String[] args)throws Exception {
        String sDate1="31/12/1998";
        Date date1=new SimpleDateFormat("dd/MM/yyyy").parse(sDate1);
System.out.println(sDate1+"\t"+date1);
}
}
```

```
31/12/1998  Thu Dec 31 00:00:00 GMT 1998
```

10) How to convert Date to String in Java

```
import  java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Calendar;
 public class DateToStringExample1 {
      public static void main(String args[]){
            Date date = Calendar.getInstance().getTime();
            DateFormat dateFormat = new SimpleDateFormat("yyyy-mm-dd hh:mm:ss");
            String strDate = dateFormat.format(date);
            System.out.println("Converted String: " + strDate);

      }
}
```

```
Converted String: 2021-00-26 10:00:01
```

11) How to convert String to char in Java

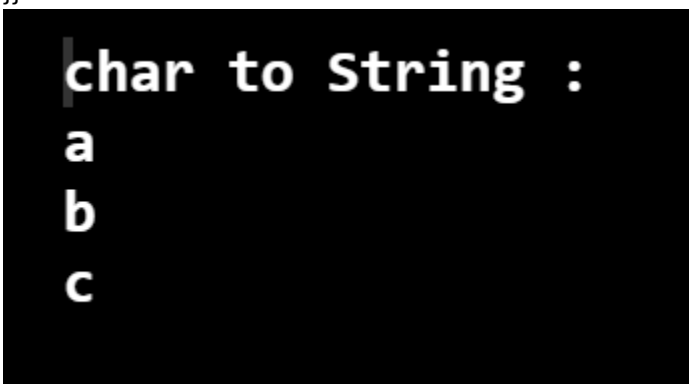
```
public class StringtoChar{
public static void main(String args[]){
System.out.println("String to char :");
          String s1="a";
          char c1=s1.charAt(0);
System.out.println(c1);
}}
```

```
String to char :
a
```

12) How to convert char to String in Java

```
public class CharTostring{
public static void main(String args[]){
System.out.println("char to String :");
          String s2=Character.toString('a');
System.out.println(s2);
          s2=String.valueOf('b');
```

```
System.out.println(s2);
        s2='c'+"";
System.out.println(s2);

}}
```

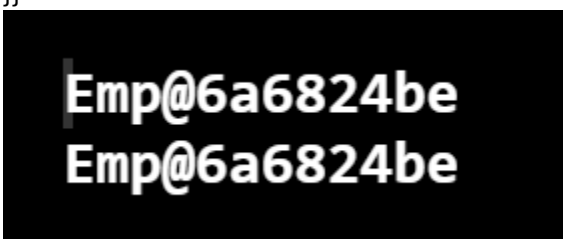


13) How to convert String to Object in Java

```
public class StringToObjectExample{
public static void main(String args[]){
String s="hello";
Object obj=s;
System.out.println(obj);
}}
```

14) How to convert Object to String in Java

```
class Emp{}
public class ObjectToStringExample{
public static void main(String args[]){
Emp e=new Emp();
String s=e.toString();
String s2=String.valueOf(e);
System.out.println(s);
System.out.println(s2);
}}
```



15) How to convert int to long in Java

```
public class intTolong{
```

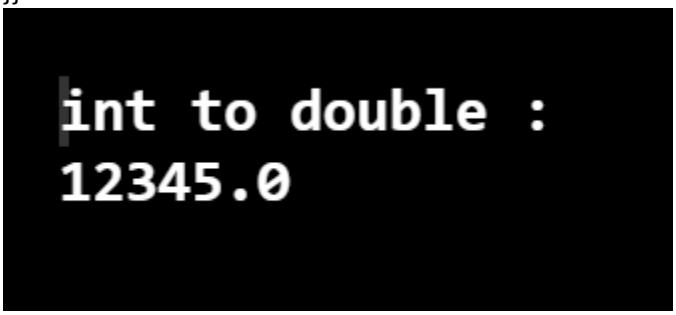
```
public static void main(String args[]){
System.out.println("int to long :");
long l=(long)58699;
System.out.println(l);
}}
```

16) How to convert long to int in Java

```
public class longToint{
public static void main(String args[]){
System.out.println("long to int :");
        int i2=(int)12345678L;
System.out.println(i2);
}}
```

17) How to convert int to double in Java

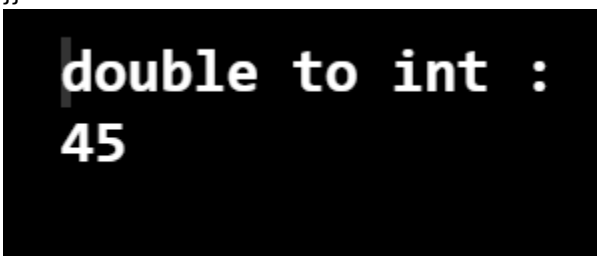
```
public class intToDouble{
public static void main(String args[]){
System.out.println("int to double :");
        double d2=(double)12345;
System.out.println(d2);
}}
```



18) How to convert double to int in Java

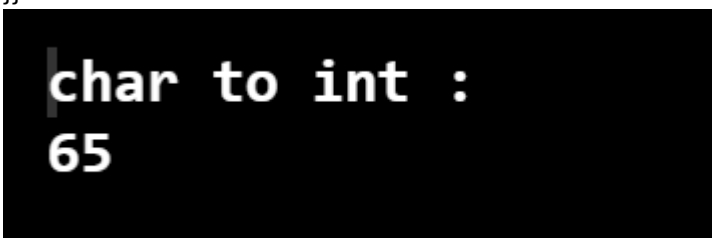
```
public class DoubleToInt{
```

```
public static void main(String args[]){
System.out.println("double to int :");
        int i3=(int)45.44;
System.out.println(i3);
}}
```



19) How to convert char to int in Java

```
public class CharToInt{
public static void main(String args[]){
System.out.println("char to int :");
        int i4=(int)'A';
System.out.println(i4);
}}
```



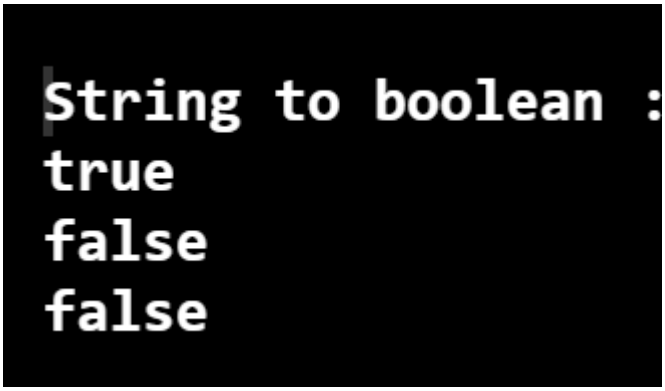
20) How to convert int to char in Java

```
public class IntToChar{
public static void main(String args[]){
System.out.println("int to char :");
        char c3=(char)65;
System.out.println(c3);
}}
```

21) How to convert String to boolean in Java

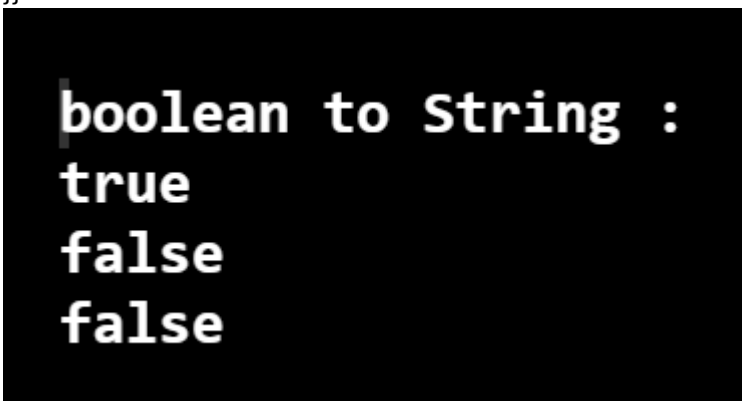
```
public class StringtoBoolean{
public static void main(String args[]){
System.out.println("String to boolean :");
boolean bool=Boolean.parseBoolean("true");
```

```
System.out.println(bool);
        bool=Boolean.parseBoolean("false");
System.out.println(bool);
            bool=Boolean.parseBoolean("abc");
System.out.println(bool);
}}
```



22) How to convert boolean to String in Java

```
public class BooleantoString{
public static void main(String args[]){
System.out.println("boolean to String :");
            String s3=Boolean.toString(true);
System.out.println(s3);
            s3=String.valueOf(false);
System.out.println(s3);
            s3=false+"";
System.out.println(s3);
}}
```



23) How to convertDate to Timestamp in Java

```
import java.sql.Timestamp;
import java.util.Date;
 public class DateToTimestamp {
            public static void main(String args[]){
            Date date = new Date();
```

```
        Timestamp ts=new Timestamp(date.getTime());
System.out.println(ts);
}
}
```

```
2021-11-26 10:20:28.68
```

24) How to convertTimestamp to Date in Java

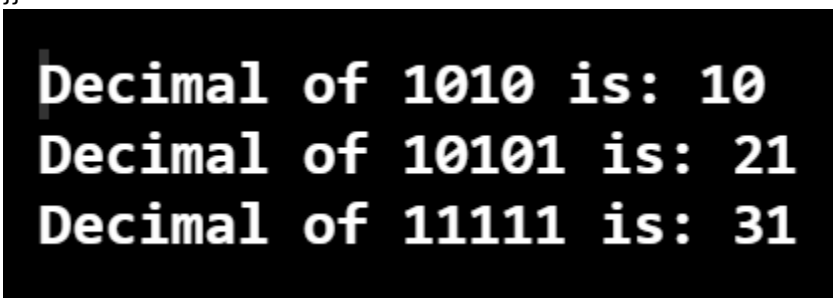
```
import java.sql.Timestamp;
import java.util.Date;
 public class TimestampToDateExample1 {
        public static void main(String args[]){
        Timestamp ts=new Timestamp(System.currentTimeMillis());
        Date date=new Date(ts.getTime());
System.out.println(date);
        }
}
```

```
Fri Nov 26 10:20:46 GMT 2021
```

25) How to convertBinary to Decimal in Java

```
public class BinaryToDecimalExample3{
public static int getDecimal(int binary){
        int decimal = 0;
        int n = 0;
        while(true){
if(binary == 0){
        break;
        } else {
        int temp = binary%10;
        decimal += temp*Math.pow(2, n);
        binary = binary/10;
        n++;
        }
        }
        return decimal;
}
public static void main(String args[]){
System.out.println("Decimal of 1010 is: "+getDecimal(1010));
System.out.println("Decimal of 10101 is: "+getDecimal(10101));
```

```
System.out.println("Decimal of 11111 is: "+getDecimal(11111));
}}
```



26) How to convertDecimal to Binary in Java

```
public class DecimalToBinary2{
public static void toBinary(int decimal){
        int binary[] = new int[40];
        int index = 0;
while(decimal > 0){
        binary[index++] = decimal%2;
        decimal = decimal/2;
        }
for(int i = index-1;i >= 0;i--){
System.out.print(binary[i]);
        }
System.out.println();
}
public static void main(String args[]){
System.out.println("Decimal of 10 is: ");
toBinary(10);
System.out.println("Decimal of 21 is: ");
toBinary(21);
System.out.println("Decimal of 31 is: ");
toBinary(31);
}}
```

```
Decimal of 10 is:
1010
Decimal of 21 is:
10101
Decimal of 31 is:
11111
```

27) How to convertHex to Decimal in Java

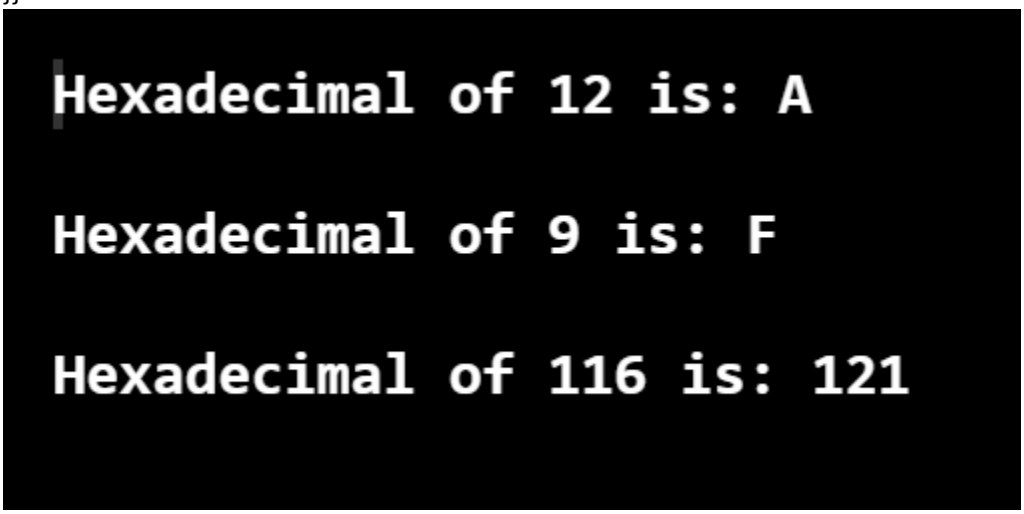
```
public class HexToDecimal{
public static int getDecimal(String hex){
        String digits = "0123456789ABCDEF";
        hex = hex.toUpperCase();
        int val = 0;
        for (int i = 0; i<hex.length(); i++)
        {
        char c = hex.charAt(i);
        int d = digits.indexOf(c);
val = 16*val + d;
        }
        return val;
}
public static void main(String args[]){
System.out.println("Decimal of b is: "+getDecimal("b"));
System.out.println("Decimal of e is: "+getDecimal("e"));
System.out.println("Decimal of 451 is: "+getDecimal("451"));
}}
```

```
Decimal of b is: 11
Decimal of e is: 14
Decimal of 451 is: 1105
```

28) How to convertDecimal to Hex in Java

```
public class DecimalToHex{
public static String toHex(int decimal){
        int rem;
```

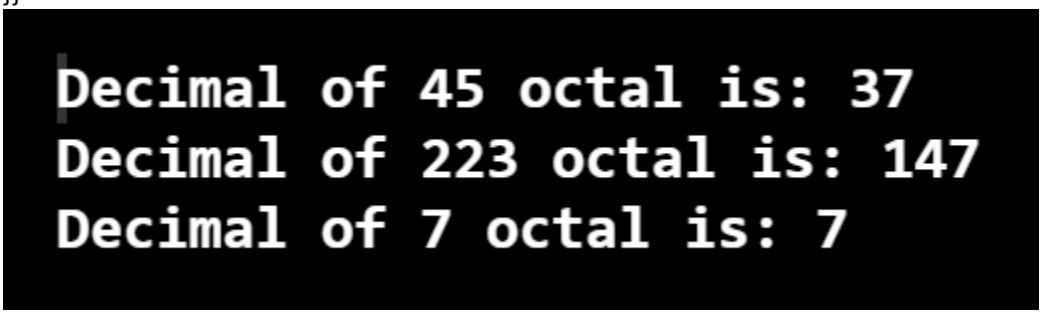
```
String hex="";
        char hexchars[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
        while(decimal>0)
        {
        rem=decimal%16;
        hex=hexchars[rem]+hex;
        decimal=decimal/16;
        }
        return hex;
}
public static void main(String args[]){
System.out.println("Hexadecimal of 12 is: "+toHex(10)+"\n");
System.out.println("Hexadecimal of 9 is: "+toHex(15)+"\n");
System.out.println("Hexadecimal of 116 is: "+toHex(289));
}}
```



29) How to convertOctal to Decimal in Java

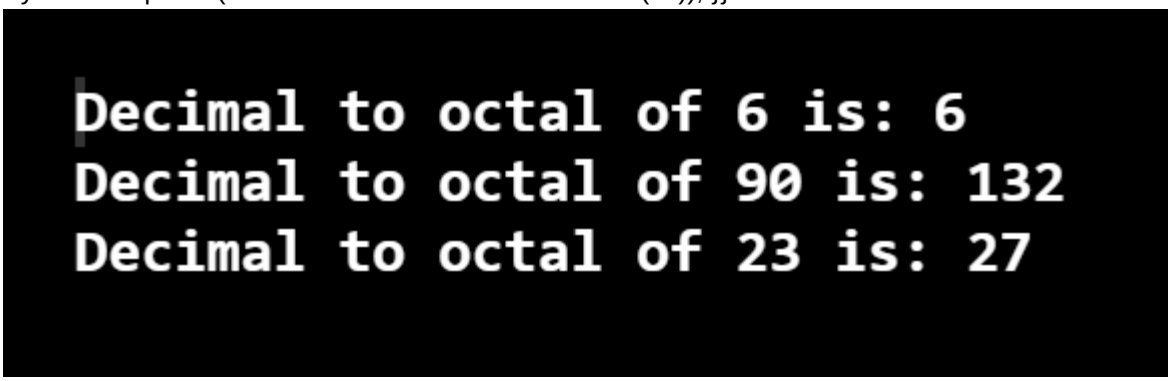
```
public class OctalToDecimal{
public static int getDecimal(int octal){
        int decimal = 0;
   int n = 0;
        while(true){
if(octal == 0){
        break;
        } else {
        int temp = octal%10;
        decimal += temp*Math.pow(8, n);
        octal = octal/10;
        n++;
        }
        }
        return decimal;
}
public static void main(String args[]){
```

```
System.out.println("Decimal of 45 octal is: "+getDecimal(45));
System.out.println("Decimal of 223 octal is: "+getDecimal(223));
System.out.println("Decimal of 7 octal is: "+getDecimal(7));
}}
```



30) How to convertDecimal to Octal in Java

```
public class DecimalToOctal{
public static String toOctal(int decimal){
        int rem;
        String octal="";
        char octalchars[]={'0','1','2','3','4','5','6','7'};
        while(decimal>0)
        {
        rem=decimal%8;
        octal=octalchars[rem]+octal;
    decimal=decimal/8;
        }
        return octal;
}
public static void main(String args[]){
System.out.println("Decimal to octal of 6 is: "+toOctal(6));
System.out.println("Decimal to octal of 90 is: "+toOctal(90));
System.out.println("Decimal to octal of 23 is: "+toOctal(23)); }}
```



Java Singly Linked List Programs

1) Singly linked list Examples in Java

```
public class Main {

	public Node head = null;
	class Node {
		private int data;
		private Node next;

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

	public void addNodeAtTheBeginning(int data) {
	System.out.println("Add a node with data " + data + " in the beginning.");
		Node newNode = new Node(data);

		if (this.head == null)
			this.head = newNode;
		} else {
			newNode.next = this.head;

			this.head = newNode;
		}
	}

	public void addNodeAtTheEnd(int data) {
	System.out.println("Add a node with data " + data + " at the end.");
		Node newNode = new Node(data);

		if (this.head == null)
			this.head = newNode;
		} else {
			Node cur = this.head;
			// traverse to the end of the list
			while (cur.next != null) {
				cur = cur.next;
			}
			cur.next = newNode;
		}
	}

	public void add(int position, int data) {
	System.out.println("Add a node with data " + data + " at the position " + position);

		Node newNode = new Node(data);
```

```
		Node cur = this.head, prev = this.head;

		if (position == 1) {
			// Point the new node's next to head
			newNode.next = head;
			// Make the new node as head
			this.head = newNode;
			return;
		}

		while (cur.next != null && --position > 0) {

			prev  = cur;
			cur = cur.next;
		}

		prev.next = newNode;

		newNode.next = cur;
	}

	public void print() {
		if (this.head == null) {
			System.out.println("The List is empty.");
		} else {
			System.out.println("The contents of the Singly Linked List are : ");
			Node cur = this.head;
			while (cur != null) {
				System.out.print(cur.data + " -> ");
				cur = cur.next;
			}
			System.out.println("NULL\n");
		}
	}

	public static void main(String[] args) {
Main list = new Main();
System.out.println("Created a singly linked list.....");
	list.print();
		list.addNodeAtTheBeginning(100);
	list.print();
		list.addNodeAtTheBeginning(200);
		list.print();
	list.addNodeAtTheEnd(900);
	list.print();
		list.addNodeAtTheEnd(800);
		list.print();
list.add(1,150);
```

```
        list.print();
        list.add(4,250);
        list.print();
        list.add(6,250);
        list.print();
}
}
```

```
Created a singly linked list .....
The List is empty.
Add a node with data 100 in the beginning.
The contents of the Singly Linked List are :
100 -> NULL

Add a node with data 200 in the beginning.
The contents of the Singly Linked List are :
200 -> 100 -> NULL

Add a node with data 900 at the end.
The contents of the Singly Linked List are :
200 -> 100 -> 900 -> NULL

Add a node with data 800 at the end.
The contents of the Singly Linked List are :
200 -> 100 -> 900 -> 800 -> NULL

Add a node with data 150 at the position 1
The contents of the Singly Linked List are :
150 -> 200 -> 100 -> 900 -> 800 -> NULL

Add a node with data 250 at the position 4
The contents of the Singly Linked List are :
150 -> 200 -> 100 -> 250 -> 900 -> 800 -> NULL

Add a node with data 250 at the position 6
The contents of the Singly Linked List are :
150 -> 200 -> 100 -> 250 -> 900 -> 250 -> 800 -> NULL
```

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

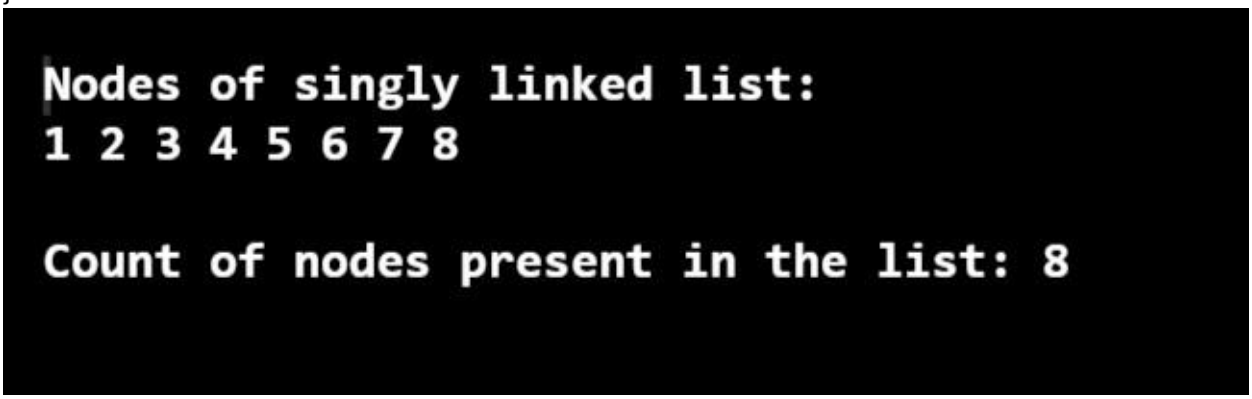
```
public class SinglyLinkedList {
        class Node{
        int data;
        Node next;
```

```
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public Node head = null;
        public Node tail = null;
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
tail.next = newNode;
        tail = newNode;
        }
        }
        public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
        return;
        }
System.out.println("Nodes of singly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
        current = current.next;
        }
System.out.println();
        }
        public static void main(String[] args) {
SinglyLinkedList sList = new SinglyLinkedList();
sList.addNode(2);
sList.addNode(3);
sList.addNode(5);
sList.addNode(0);
sList.addNode(7);
sList.display();
        }
}
```

3) Java program to create a singly linked list of n nodes and count the number of nodes.

```
public class CountNodes {
        class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public Node head = null;
        public Node tail = null;
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
tail.next = newNode;
        tail = newNode;
        }
        }
        public int countNodes() {
        int count = 0;
        Node current = head;
while(current != null) {
        count++;
        current = current.next;
        }
        return count;
        }
        public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
        return;
```

```
        }
System.out.println("Nodes of singly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
        current = current.next;
        }
System.out.println();
        }
        public static void main(String[] args) {
CountNodes sList = new CountNodes();
sList.addNode(1);
sList.addNode(2);
sList.addNode(3);
sList.addNode(4);
sList.addNode(5);
sList.addNode(6);
sList.addNode(7);
sList.addNode(8);
sList.display();
System.out.println("\r\n"+"Count of nodes present in the list: " + sList.countNodes());
        }
}
```



4) Java program to create a singly linked list of n nodes and display it in reverse order.

```
public class ReverseList {
        class Node{
        int data;
        Node next;

        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public Node head = null;
        public Node tail = null;
```

```
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
tail.next = newNode;
        tail = newNode;
        }
        }
        public void reverse(Node current) {
if(head == null) {
System.out.println("List is empty");
        return;
        }
        else {
if(current.next == null) {
System.out.print(current.data + " ");
            return;
        }
        reverse(current.next);
System.out.print(current.data + " ");
        }
        }
        public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
              return;
        }
while(current != null) {
System.out.print(current.data + " ");
        current = current.next;
        }
System.out.println();
        }
        public static void main(String[] args) {
ReverseList sList = new ReverseList();
sList.addNode(2);
sList.addNode(3);
sList.addNode(5);
sList.addNode(0);
sList.addNode(7);
System.out.println("Original List: ");
sList.display();
System.out.println("Reversed List: ");
sList.reverse(sList.head);
```

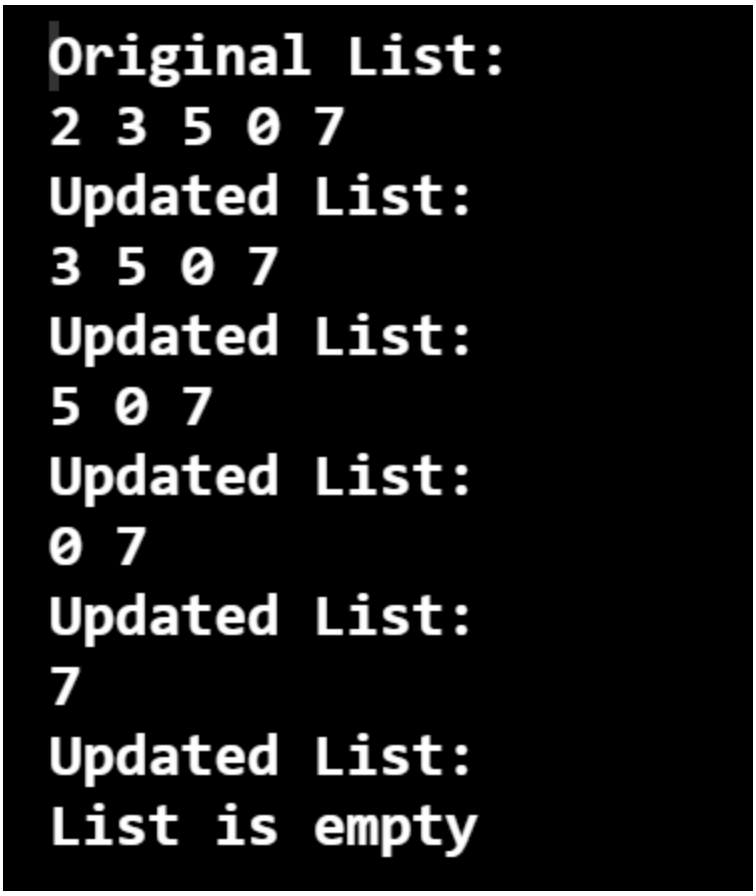
}}

```
Original List:
2 3 5 0 7
Reversed List:
7 0 5 3 2
```

5) Java program to delete a node from the beginning of the singly linked list.

```
public class DeleteStart {
        class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public Node head = null;
        public Node tail = null;
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
tail.next = newNode;
        tail = newNode;
        }
        }
        public void deleteFromStart() {
if(head == null) {
System.out.println("List is empty");
        return;
        }
        else {
if(head != tail) {
        head = head.next;
        }
        else {
        head = tail = null;
        }
```

```
        }
        }
        public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
        return;
        }
while(current != null) {
System.out.print(current.data + " ");
        current = current.next;
        }
System.out.println();
        }
        public static void main(String[] args) {
DeleteStartsList = new DeleteStart();
sList.addNode(2);
sList.addNode(3);
sList.addNode(5);
sList.addNode(0);
sList.addNode(7);
System.out.println("Original List: ");
sList.display();
while(sList.head != null) {
sList.deleteFromStart();
System.out.println("Updated List: ");
sList.display();
        }}}
```

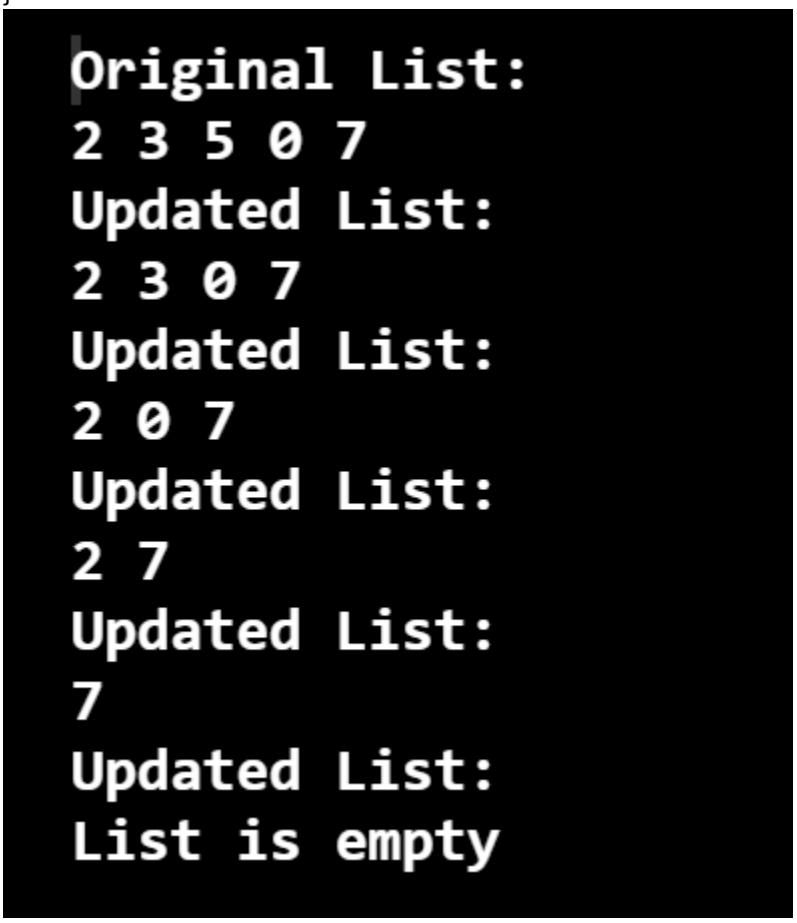


6) Java program to delete a node from the middle of the singly linked list.

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

public int size;
 public void addNode(int data) {
Node newNode = new Node(data);
if(head == null) {
head = newNode;
tail = newNode;
```

```
}
else {
tail.next = newNode;
tail = newNode;
}
size++;
}
void deleteFromMid() {
Node temp, current;
if(head == null) {
System.out.println("List is empty");
return;
}
else {
 int count = (size % 2 == 0) ? (size/2) : ((size+1)/2);
if( head != tail ) {
temp = head;
current = null;
for(int i = 0; i< count-1; i++){
current = temp;
temp = temp.next;
}
if(current != null) {
current.next = temp.next;
temp = null;
 }
 else {
head = tail = temp.next;
temp = null;
}
}
else {
head = tail = null;
}
}
size--;
}
public void display() {
Node current = head;
if(head == null) {
System.out.println("List is empty");
 return;
            }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
            }
System.out.println();
```

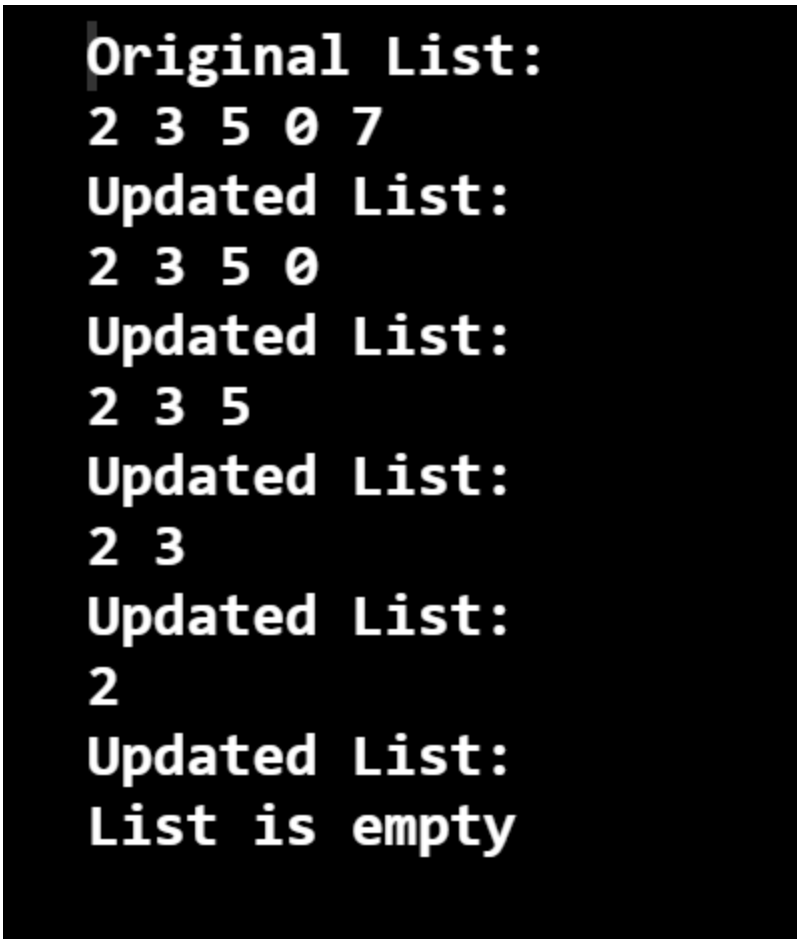
```
        }

public static void main(String[] args) {

deleteMidsList = new deleteMid();
sList.addNode(2);
sList.addNode(3);
sList.addNode(5);
sList.addNode(0);
sList.addNode(7);
System.out.println("Original List: ");
sList.display();

while(sList.head != null) {
sList.deleteFromMid();
System.out.println("Updated List: ");
sList.display();
        }
    }
}
```



7) Java program to delete a node from the end of the singly linked list

```
public class DeleteEnd {
   class Node{
            int data;
            Node next;

            public Node(int data) {
this.data = data;
this.next = null;
            }
            }
            public Node head = null;
            public Node tail = null;
            public void addNode(int data) {
              Node newNode = new Node(data);
if(head == null) {
            head = newNode;
            tail = newNode;
            }
            else {
tail.next = newNode;
            tail = newNode;
            }
            }
            public void deleteFromEnd() {
if(head == null) {
System.out.println("List is empty");
            return;
            }
            else {
if(head != tail ) {
            Node current = head;
while(current.next != tail) {
                  current = current.next;
            }
            tail = current;
tail.next = null;
            }
            else {
            head = tail = null;
            }
            }
            }
            public void display() {
            Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
            }
```

```
while(current != null) {
System.out.print(current.data + " ");
        current = current.next;
        }
System.out.println();
        }
        public static void main(String[] args) {
DeleteEnd sList = new DeleteEnd();
sList.addNode(2);
sList.addNode(3);
sList.addNode(5);
sList.addNode(0);
sList.addNode(7);
System.out.println("Original List: ");
sList.display();
while(sList.head != null) {
sList.deleteFromEnd();
System.out.println("Updated List: ");
sList.display();
        }
        }
}
```

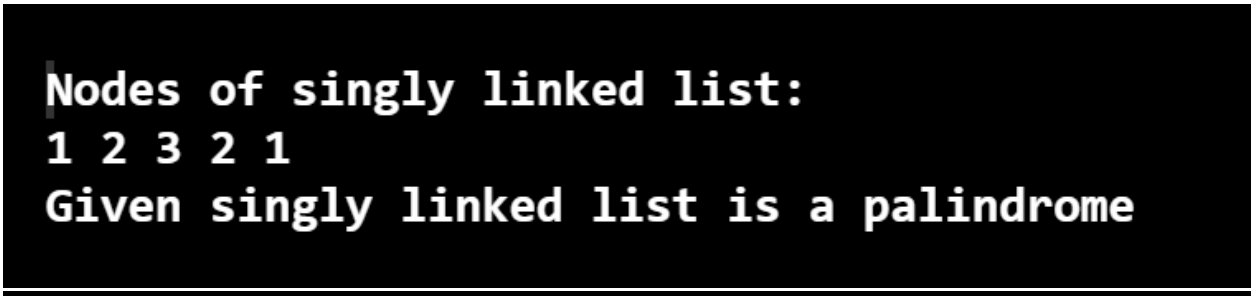


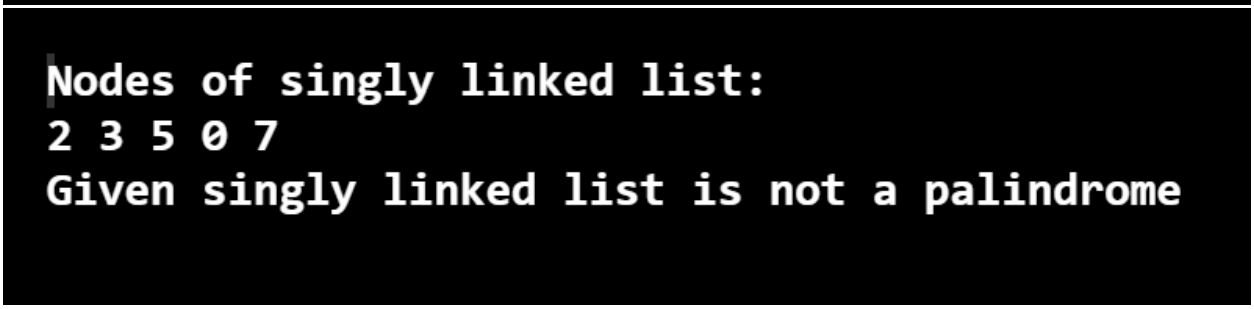
8) Java program to determine whether a singly linked list is the palindrome.

```
public class PalindromeLL {
        class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public int size;
        public Node head = null;
        public Node tail = null;
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
```

```
            else {
tail.next = newNode;
            tail = newNode;
            }
            size++;
            }
            public Node reverseList(Node temp){
            Node current = temp;
            Node prevNode = null, nextNode = null;
while(current != null){
nextNode = current.next;
current.next = prevNode;
prevNode = current;
            current = nextNode;
            }
            return prevNode;
            }
            public void isPalindromeLL(){
            Node current = head;
boolean flag = true;
            int mid = (size%2 == 0)? (size/2) : ((size+1)/2);
for(int i=1; i<mid; i++){
            current = current.next;
            }
            Node revHead = reverseList(current.next);
while(head != null &&revHead != null){
if(head.data != revHead.data){
            flag = false;
            break;
                }
            head = head.next;
revHead = revHead.next;
            }
            if(flag)
System.out.println("Given singly linked list is a palindrome");
            else
System.out.println("Given singly linked list is not a palindrome");
            }
            public void display() {
            Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
            }
System.out.println("Nodes of singly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
```

```
        }
System.out.println();
        }
        public static void main(String[] args) {
PalindromeLL sList = new PalindromeLL();
sList.addNode(2);
sList.addNode(3);
sList.addNode(5);
sList.addNode(0);
sList.addNode(7); sList.display();
sList.isPalindromeLL();
        }
}
```





9) Java program to find the maximum and minimum value node from a linked list.

```
public class MinMax {
        class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public Node head = null;
    public Node tail = null;
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
```

```
        }
        else {
tail.next = newNode;
        tail = newNode;
        }
        }
        public void minNode() {
        Node current = head;
        int min;
if(head == null) {
System.out.println("List is empty");
        }
        else {
        min = head.data;
while(current != null){
if(min >current.data) {
        min = current.data;
        }
        current= current.next;
        }
System.out.println("Minimum value node in the list: "+ min+"\r\n");
        }
        }
        public void maxNode() {
        Node current = head;
        int max;
if(head == null) {
System.out.println("List is empty");
        }
        else {
        max = head.data;
while(current != null){
if(max <current.data) {
        max = current.data;
        }
        current = current.next;
        }
System.out.println("Maximum value node in the list: "+ max);
        }
        }
        public static void main(String[] args) {
MinMax sList = new MinMax();
sList.addNode(2);
sList.addNode(3);
sList.addNode(5);
sList.addNode(0);
sList.addNode(7);
sList.minNode();
```

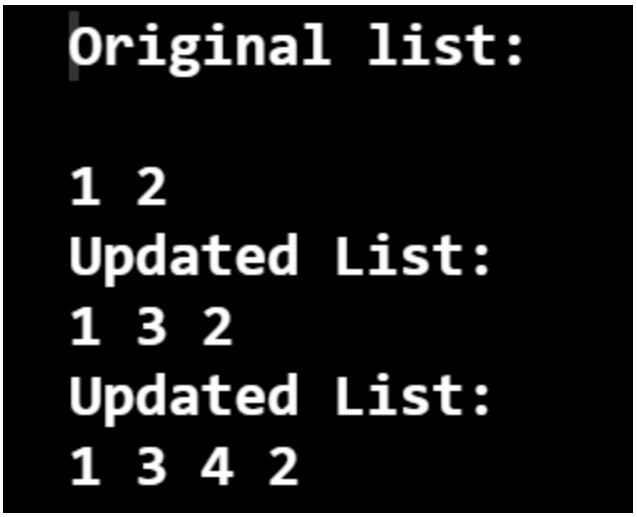
```
sList.maxNode();
        }
}
```

```
Minimum value node in the list: 0

Maximum value node in the list: 7
```

10) Java Program to insert a new node at the middle of the singly linked list.

```
public class InsertMid {
        class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public int size;
        public Node head = null;
        public Node tail = null;
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
tail.next = newNode;
        tail = newNode;
        }
        size++;
        }
        public void addInMid(int data){
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
        Node temp, current;
```

```
            int count = (size % 2 == 0) ? (size/2) : ((size+1)/2);
            temp = head;
            current = null;
for(int i = 0; i< count; i++) {
            current = temp;
            temp = temp.next;
            }
current.next = newNode;
newNode.next = temp;
            }
            size++;
            }
            public void display() {
            Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
            }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
            }
System.out.println();
            }
            public static void main(String[] args) {
InsertMid sList = new InsertMid();
sList.addNode(1);
sList.addNode(2);
System.out.println("Original list: "+"\r\n");
sList.display();
sList.addInMid(3);
System.out.println( "Updated List: ");
sList.display();
sList.addInMid(4);
System.out.println("Updated List: ");
sList.display();
            }
}
```



11) Java program to insert a new node at the beginning of the singly linked list.

```
public class InsertStart {
        class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public Node head = null;
        public Node tail = null;
        public void addAtStart(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
        Node temp = head;
        head = newNode;
head.next = temp;
        }
        }
        public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
        return;
        }
System.out.println("Adding nodes to the start of the list: ");
```

```
while(current != null) {
System.out.print(current.data + " ");
        current = current.next;
        }
System.out.println();
        }
        public static void main(String[] args) {
InsertStart sList = new InsertStart();
sList.addAtStart(4);
sList.display();
sList.addAtStart(5);
sList.display();
sList.addAtStart(6);
sList.display();
sList.addAtStart(7);
sList.display();
        }
}
```

```
Adding nodes to the start of the list:
4
Adding nodes to the start of the list:
5 4
Adding nodes to the start of the list:
6 5 4
Adding nodes to the start of the list:
7 6 5 4
```

12) Java program to insert a new node at the end of the singly linked list.

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

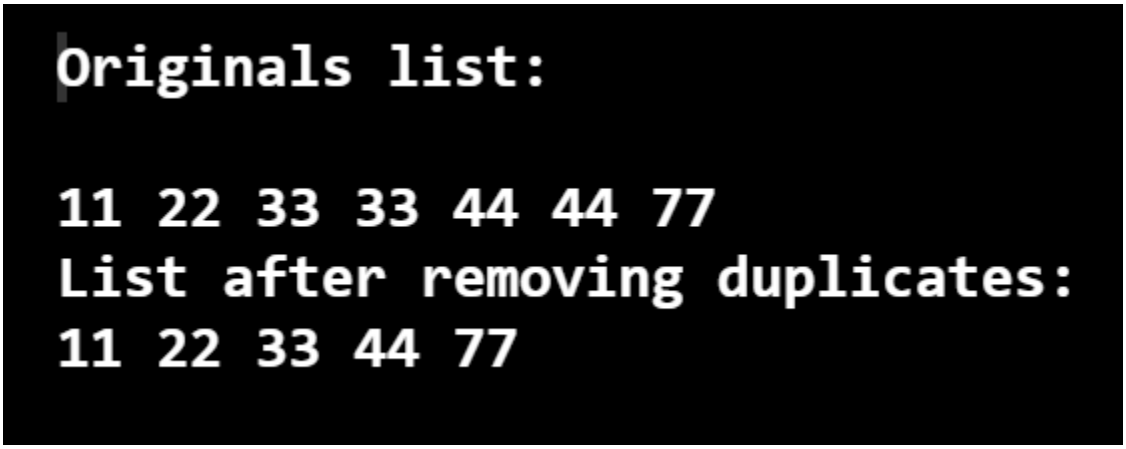
```
	public Node head = null;
	public Node tail = null;
	public void addAtEnd(int data) {
	Node newNode = new Node(data);
if(head == null) {
	head = newNode;
	tail = newNode;
	}
	else {
tail.next = newNode;
	tail = newNode;
	}
	}
	public void display() {
	Node current = head;
if(head == null) {
System.out.println("List is empty");
	return;
	}
System.out.println("Adding nodes to the end of the list: ");
while(current != null) {
System.out.print(current.data + " ");
	current = current.next;
	}
System.out.println();
	}
	public static void main(String[] args) {
InsertEnd sList = new InsertEnd();
sList.addAtEnd(5);
sList.display();
sList.addAtEnd(6);
sList.display();
sList.addAtEnd(7);
sList.display();
sList.addAtEnd(8);
sList.display();
	}
}
```

```
Adding nodes to the end of the list:
5
Adding nodes to the end of the list:
5 6
Adding nodes to the end of the list:
5 6 7
Adding nodes to the end of the list:
5 6 7 8
```

13) Java program to remove duplicate elements from a singly linked list.

```
public class RemoveDuplicate {
        class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public Node head = null;
        public Node tail = null;
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
tail.next = newNode;
        tail = newNode;
        }
        }
        public void removeDuplicate() {
        Node current = head, index = null, temp = null;
if(head == null) {
        return;
        }
        else {
```

```
while(current != null){
                temp = current;
            index = current.next;
while(index != null) {
if(current.data == index.data) {
temp.next = index.next;
            }
            else {
            temp = index;
                  }
            index = index.next;
            }
            current = current.next;
            }
            }
            }
            public void display() {
            Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
            }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
            }
System.out.println();
            }
            public static void main(String[] args) {
RemoveDuplicate sList = new RemoveDuplicate();
sList.addNode(11);
sList.addNode(22);
sList.addNode(33);
sList.addNode(33);
sList.addNode(44);
sList.addNode(44);
sList.addNode(77);
System.out.println("Originals list: "+"\r\n");
sList.display();
sList.removeDuplicate();
System.out.println("List after removing duplicates: ");
sList.display();
            }
}
```



14) Java Program to search an element in a singly linked list.

```
public class SearchLinkedList {
        class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
this.next = null;
        }
        }
        public Node head = null;
        public Node tail = null;
        public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
        }
        else {
tail.next = newNode;
        tail = newNode;
        }
        }
        public void searchNode(int data) {
        Node current = head;
        int i = 1;
boolean flag = false;
if(head == null) {
System.out.println("List is empty");
        }
        else {
while(current != null) {
if(current.data == data) {
```

```
            flag = true;
            break;
            }
i++;
            current = current.next;
            }
            }
            if(flag)
System.out.println("Element is present in the list at the position : " + i+"\n");
            else
System.out.println("Element is not present in the list");
            }
            public static void main(String[] args) {
SearchLinkedList sList = new SearchLinkedList();
sList.addNode(23);
sList.addNode(24);
sList.addNode(32);
sList.addNode(42);
sList.searchNode(23);
sList.searchNode(71);
            }}
```

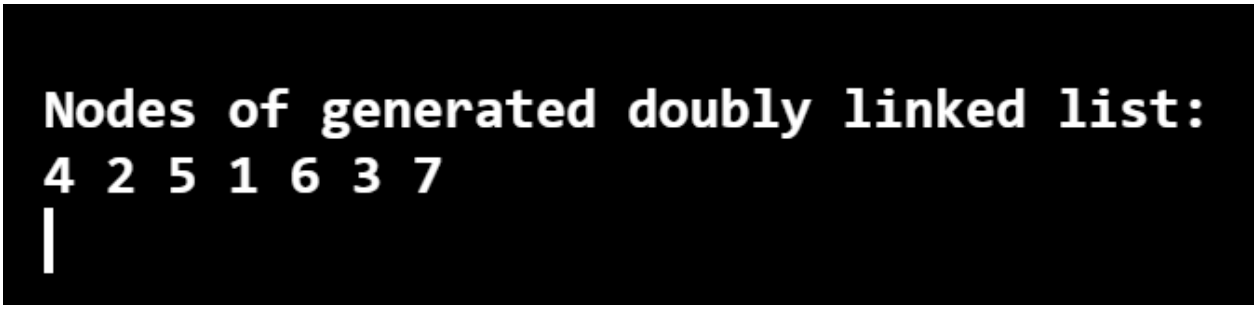
```
Element is present in the list at the position : 1

Element is not present in the list
```

Java Doubly Linked List Programs

1) Java program to convert a given binary tree to doubly linked list

```
public class BinaryTreeToDLL {
    public static class Node{
        int data;
        Node left;
        Node right;

        public Node(int data) {
this.data = data;
```

```
this.left = null;
this.right = null;
        }
    }
    public Node root;
    Node head, tail = null;
    public void convertbtToDLL(Node node) {
if(node == null)
            return;
convertbtToDLL(node.left);
if(head == null) {
            head = tail = node;
        }
        else {
tail.right = node;
node.left = tail;
            tail = node;
        }
convertbtToDLL(node.right);
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
System.out.println("Nodes of generated doubly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
            current = current.right;
        }
System.out.println();
    }
    public static void main(String[] args) {
BinaryTreeToDLL bt = new BinaryTreeToDLL();
bt.root = new Node(1);
bt.root.left = new Node(2);
bt.root.right = new Node(3);
bt.root.left.left = new Node(4);
bt.root.left.right = new Node(5);
bt.root.right.left = new Node(6);
bt.root.right.right = new Node(7);
bt.convertbtToDLL(bt.root);
bt.display();
    }
}
```



2) Java program to create a doubly linked list from a ternary tree.

```
public class TernaryTreeToDLL {
    public static class Node{
        int data;
        Node left;
        Node middle;
        Node right;

        public Node(int data) {
this.data = data;
        }
    }
    public Node root;
    Node head, tail = null;
    public void convertTernaryToDLL(Node node) {
if(node == null)
            return;
        Node left = node.left;
        Node middle = node.middle;
        Node right = node.right;
if(head == null) {
            head = tail = node;
node.middle = null;
head.left = null;
tail.right = null;
        }
        else  {
tail.right = node;
node.left = tail;
node.middle = null;
            tail = node;
tail.right = null;
        }
convertTernaryToDLL(left);
convertTernaryToDLL(middle);
convertTernaryToDLL(right);
    }
```

```
    public void displayDLL() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
System.out.println("Nodes of generated doubly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
            current = current.right;
        }
System.out.println();
    }
    public static void main(String[] args) {
TernaryTreeToDLL tree = new TernaryTreeToDLL();
tree.root = new Node(5);
tree.root.left = new Node(10);
tree.root.middle = new Node(12);
tree.root.right = new Node(15);
tree.root.left.left = new Node(20);
tree.root.left.middle = new Node(40);
tree.root.left.right = new Node(50);
tree.root.middle.left = new Node(24);
tree.root.middle.middle = new Node(36);
tree.root.middle.right = new Node(48);
tree.root.right.left = new Node(30);
tree.root.right.middle = new Node(45);
tree.root.right.right = new Node(60);
tree.convertTernaryToDLL(tree.root);
tree.displayDLL();
    }
}
```

```
Nodes of generated doubly linked list:
5 10 20 40 50 12 24 36 48 15 30 45 60
```

3) Java program to create a doubly linked list of n nodes and count the number of nodes

```
public class CountList {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
```

```
this.data = data;
        }
    }
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public int countNodes() {
        int counter = 0;
        Node current = head;
while(current != null) {
            counter++;
            current = current.next;
        }
        return counter;
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
System.out.println("Nodes of doubly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
    }
    public static void main(String[] args) {
CountList dList = new CountList();
dList.addNode(5);
dList.addNode(2);
dList.addNode(6);
dList.addNode(9);
dList.addNode(1);
dList.display();
System.out.println("\nCount of nodes present in the list: " + dList.countNodes());
```

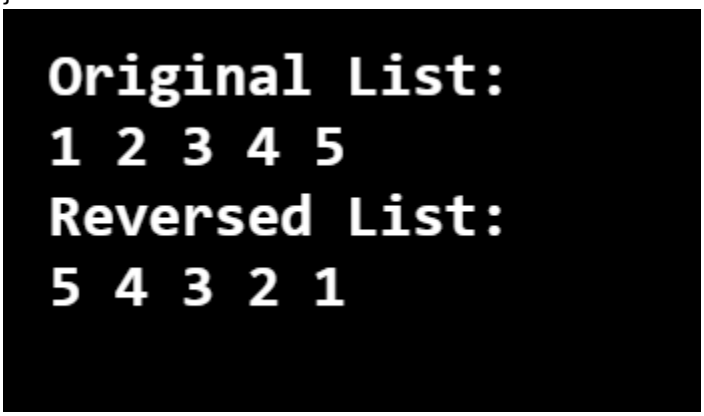
```
    }
}
```

```
Nodes of doubly linked list:
5 2 6 9 1
Count of nodes present in the list: 5
```

4) Java program to create a doubly linked list of n nodes and display it in reverse order.

```
public class ReverseList {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public void reverse() {
        Node current = head, temp = null;
while(current != null) {
            temp = current.next;
current.next = current.previous;
current.previous = temp;
            current = current.previous;
        }
        temp = head;
        head = tail;
        tail = temp;
    }
```

```
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
    }
    public static void main(String[] args) {
ReverseList dList = new ReverseList();
dList.addNode(1);
dList.addNode(2);
dList.addNode(3);
dList.addNode(4);
dList.addNode(5);
System.out.println("Original List: ");
dList.display();
dList.reverse();
System.out.println("\nReversed List: ");
dList.display();
    }
}
```



5) Java program to create and display a doubly linked list

```
public class DoublyLinkedList {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
```

```
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
System.out.println("Nodes of doubly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
    }
    public static void main(String[] args) {
DoublyLinkedList dList = new DoublyLinkedList();
dList.addNode(2);
dList.addNode(4);
dList.addNode(6);
dList.addNode(8);
dList.addNode(9);
dList.display();
    }
}
```

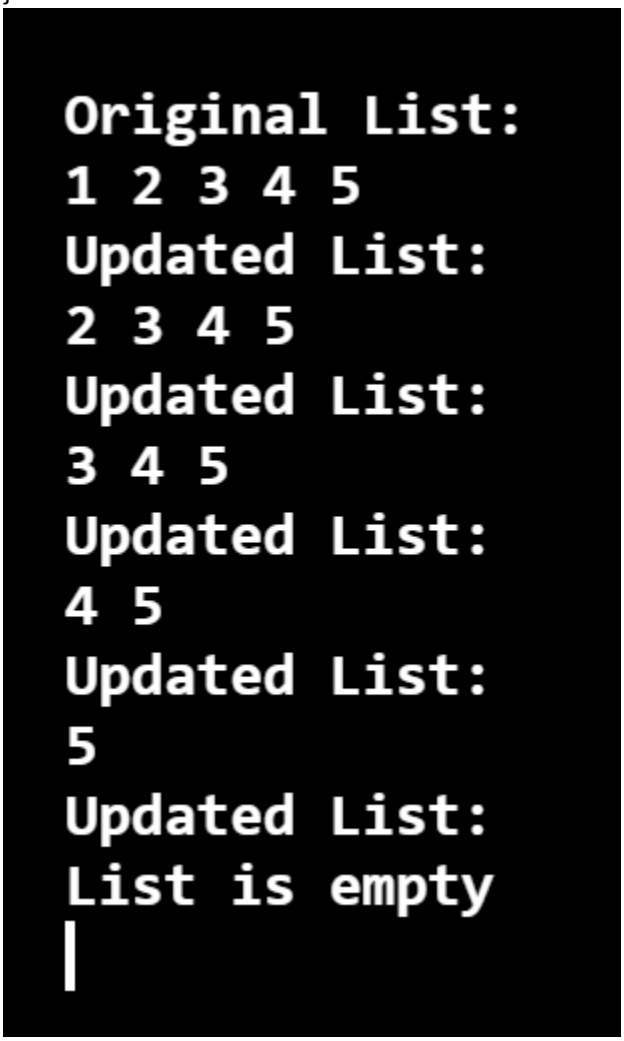
```
Nodes of doubly linked list:
2 4 6 8 9
```

6) Java program to delete a new node from the beginning of the doubly linked list.

```
public class DeleteStart
{
```

```
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
}
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public void deleteFromStart() {
if(head == null) {
            return;
        }
        else {
if(head != tail) {
                head = head.next;
head.previous = null;
            }
            else {
                head = tail = null;
            }
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
System.out.println();
```

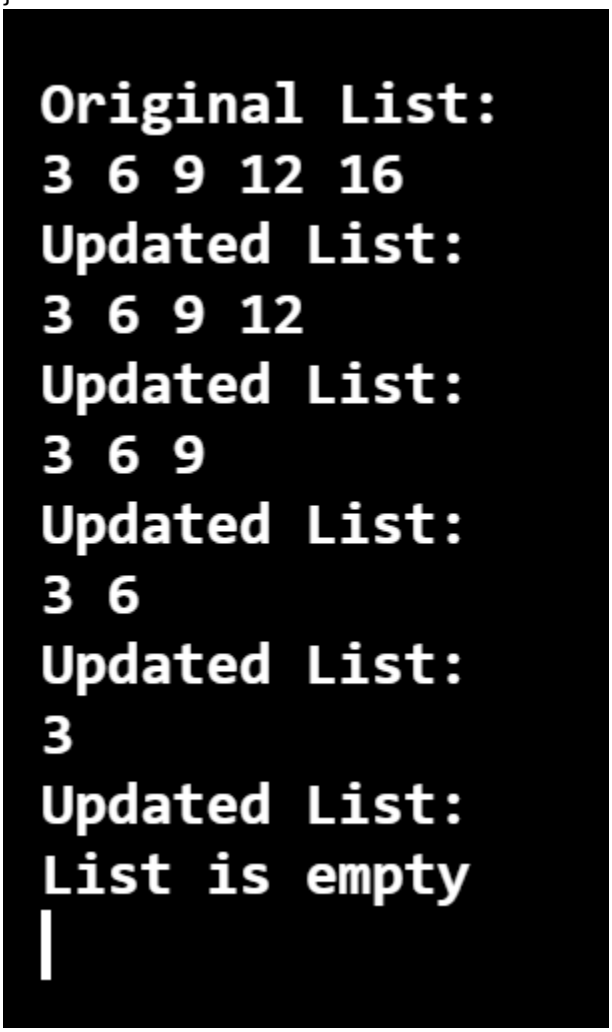
```
    }
    public static void main(String[] args) {
DeleteStart dList = new DeleteStart();
dList.addNode(1);
dList.addNode(2);
dList.addNode(3);
dList.addNode(4);
dList.addNode(5);
System.out.println("Original List: ");
dList.display();
while(dList.head != null) {
dList.deleteFromStart();
System.out.println("Updated List: ");
dList.display();
        }
    }
}
```



7) Java program to delete a new node from the end of the doubly linked list.

```
public class DeleteEnd {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public void deleteFromEnd() {
if(head == null) {
            return;
        }
        else {
if(head != tail) {
                tail = tail.previous;
tail.next = null;
            }
            else {
                head = tail = null;
            }
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
```

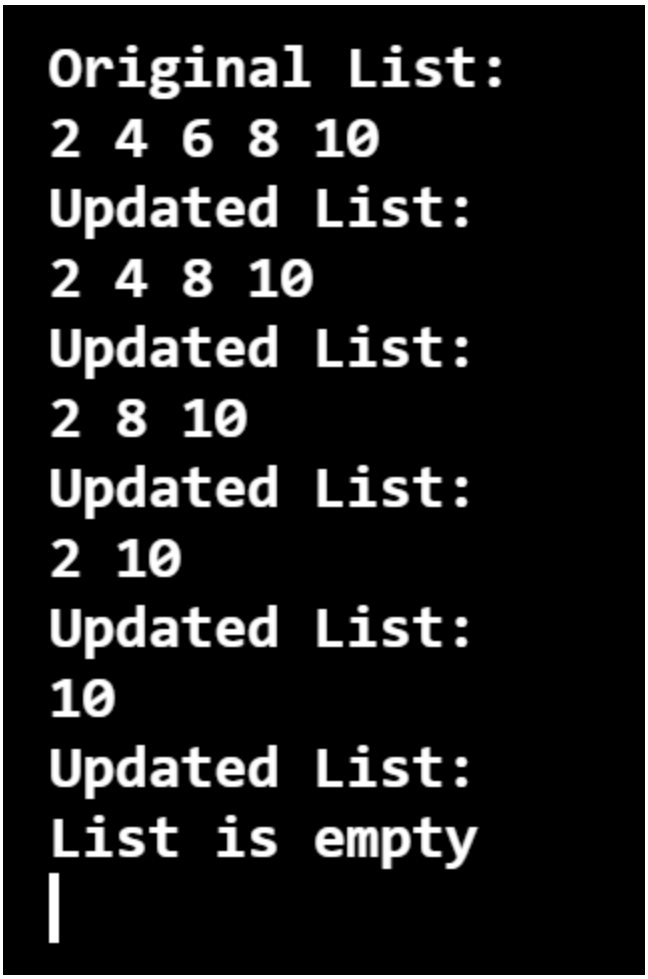
```
System.out.println();
    }
    public static void main(String[] args) {
DeleteEnd dList = new DeleteEnd();
dList.addNode(3);
dList.addNode(6);
dList.addNode(9);
dList.addNode(12);
dList.addNode(16);
System.out.println("Original List: ");
dList.display();
while(dList.head != null) {
dList.deleteFromEnd();
System.out.println("Updated List: ");
dList.display();
        }
    }
}
```



8) Java program to delete a new node from the middle of the doubly linked list.

```
public class DeleteMid {
    class Node{
        int data;
        Node previous;
        Node next;

        public Node(int data) {
this.data = data;
        }
    }
    public int size = 0;
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
        size++;
    }
    public void deleteFromMid() {
if(head == null) {
            return;
        }
        else {
            Node current = head;
            int mid = (size % 2 == 0) ? (size/2) : ((size+1)/2);
for(int i = 1; i< mid; i++){
                current = current.next;
            }
if(current == head) {
                head = current.next;
            }
            else if(current == tail) {
                tail = tail.previous;
            }
            else {
current.previous.next = current.next;
current.next.previous = current.previous;
```

```
            }
            current = null;
        }
        size--;
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
System.out.println();
    }
    public static void main(String[] args) {
DeleteMid dList = new DeleteMid();
dList.addNode(2);
dList.addNode(4);
dList.addNode(6);
dList.addNode(8);
dList.addNode(10);
System.out.println("Original List: ");
dList.display();
while(dList.head != null) {
dList.deleteFromMid();
System.out.println("Updated List: ");
dList.display();
        }
    }
}
```

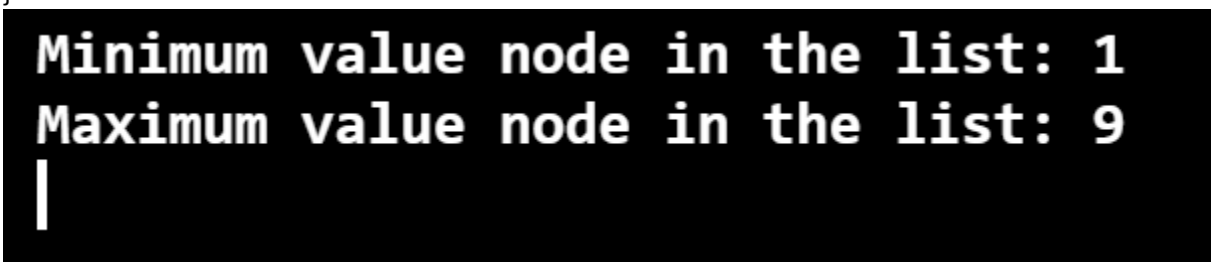


9) Java program to find the maximum and minimum value node from a doubly linked list.

```
public class MinMax {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
```

```
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public int minimumNode() {
        Node current = head;
        int min;
if(head == null) {
System.out.println("List is empty");
            return 0;
        }
        else {
            min = head.data;
while(current != null) {
if(min >current.data)
                    min = current.data;
                current = current.next;
            }
        }
        return min;
    }
    public int maximumNode() {
        Node current = head;
        int max;
if(head == null) {
System.out.println("List is empty");
            return 0;
        }
        else {
            max = head.data;
while(current != null) {
if(current.data> max)
                    max = current.data;
                current = current.next;
            }
        }
        return max;
    }
    public static void main(String[] args) {
MinMax dList = new MinMax();
dList.addNode(5);
dList.addNode(7);
dList.addNode(9);
dList.addNode(1);
dList.addNode(2);
```

```
System.out.println("Minimum value node in the list: "+ dList.minimumNode());
System.out.println("Maximum value node in the list: "+ dList.maximumNode());
    }
}
```



10) Java program to insert a new node at the beginning of the Doubly Linked list.

```
public class InsertStart {
    class Node{
        int data;
        Node previous;
        Node next;

        public Node(int data) {
this.data = data;
        }
    }
    Node head, tail = null;
    public void addAtStart(int data) {

        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
head.previous = newNode;
newNode.next = head;
newNode.previous = null;
            head = newNode;
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
System.out.println("Adding a node to the start of the list: ");
```

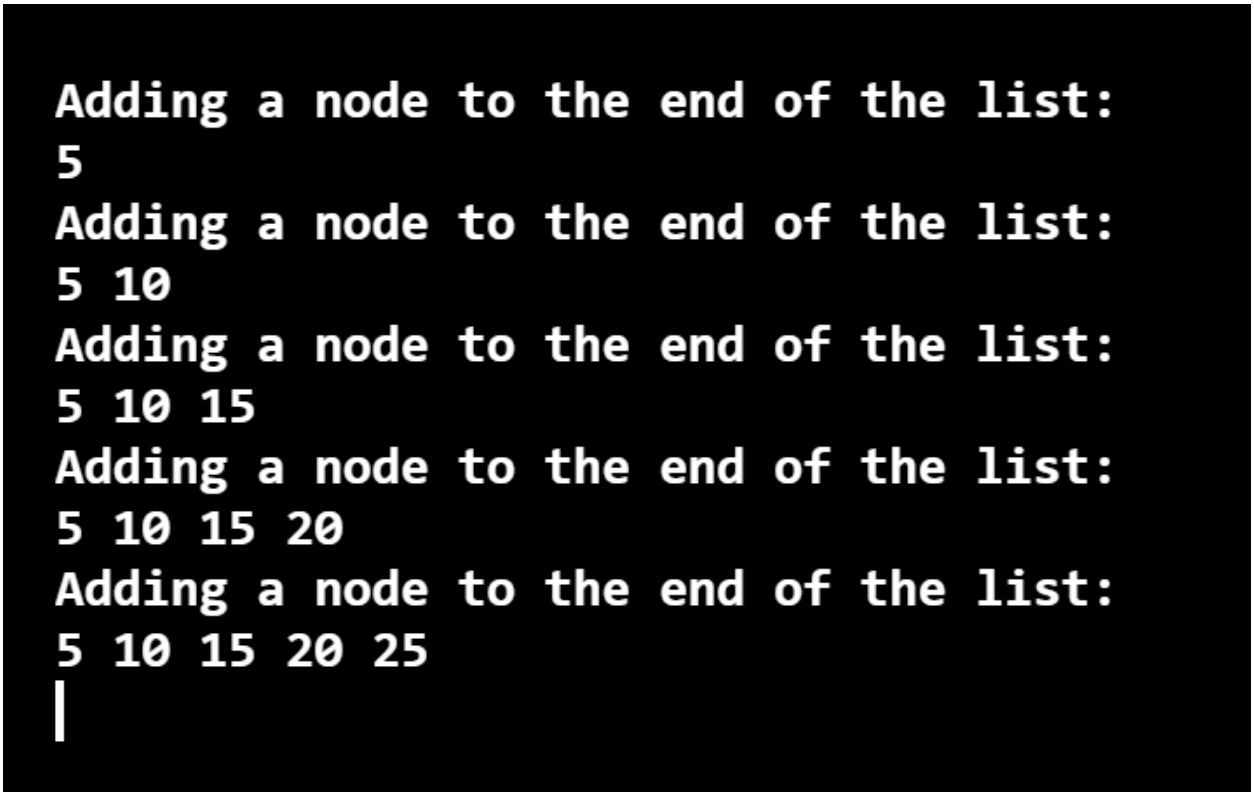
```
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
System.out.println();
    }
    public static void main(String[] args) {
InsertStart dList = new InsertStart();
dList.addAtStart(4);
dList.display();
dList.addAtStart(8);
dList.display();
dList.addAtStart(12);
dList.display();
dList.addAtStart(16);
dList.display();
dList.addAtStart(20);
dList.display();
    }
}
```

```
Adding a node to the start of the list:
4
Adding a node to the start of the list:
8 4
Adding a node to the start of the list:
12 8 4
Adding a node to the start of the list:
16 12 8 4
Adding a node to the start of the list:
20 12 16 8 4
```

11) Java program to insert a new node at the end of the Doubly Linked List.

```
public class InsertEnd {
    class Node{
        int data;
```

```
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    Node head, tail = null;
    public void addAtEnd(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
System.out.println("Adding a node to the end of the list: ");
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
System.out.println();
    }
    public static void main(String[] args) {
InsertEnd dList = new InsertEnd();
dList.addAtEnd(5);
dList.display();
dList.addAtEnd(10);
dList.display();
dList.addAtEnd(15);
dList.display();
dList.addAtEnd(20);
dList.display();
dList.addAtEnd(25);
dList.display();
    }
}
```

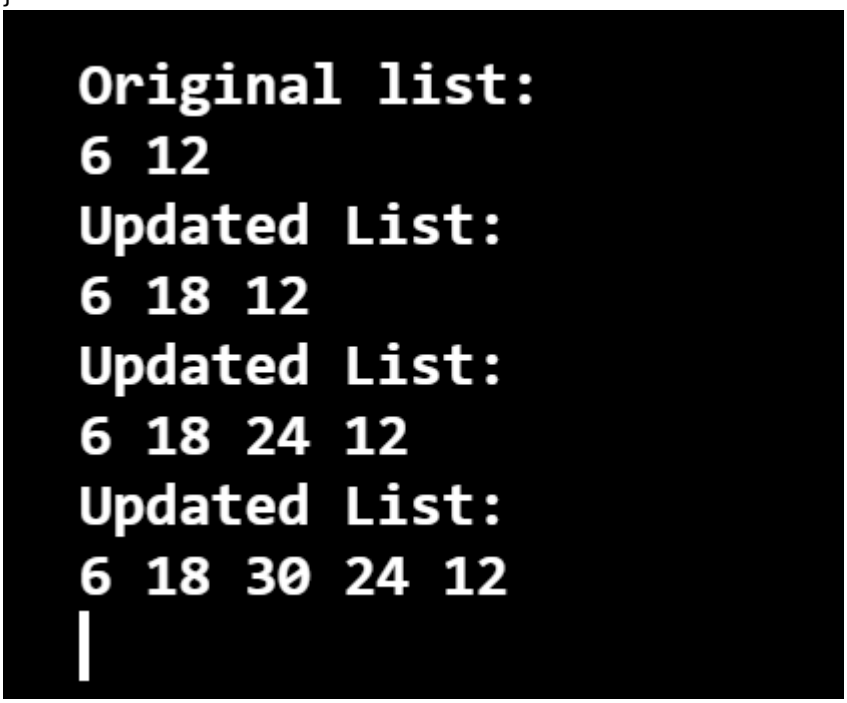


12) Java program to insert a new node at the middle of the Doubly Linked List.

```
public class InsertMid {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    public int size = 0;
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
```

```
tail.next = null;
        }
        size++;
    }
    public void addInMid(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
            Node current = head, temp = null;
            int mid = (size % 2 == 0) ? (size/2) : ((size+1)/2);
for(int i = 1; i< mid; i++){
                current = current.next;
            }
            temp = current.next;
temp.previous = current;
current.next = newNode;
newNode.previous = current;
newNode.next = temp;
temp.previous = newNode;
        }
        size++;
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
System.out.println();
    }
    public static void main(String[] args) {
InsertMid dList = new InsertMid();
dList.addNode(6);
dList.addNode(12);
System.out.println("Original list: ");
dList.display();
dList.addInMid(18);
System.out.println( "Updated List: ");
dList.display();
dList.addInMid(24);
```

```
System.out.println("Updated List: ");
dList.display();
dList.addInMid(30);
System.out.println("Updated List: ");
dList.display();
    }
}
```



13) Java program to remove duplicate elements from a Doubly Linked List.

```
public class RemoveDuplicate {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
```

```
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public void removeDuplicateNode() {
        Node current, index, temp;
if(head == null) {
            return;
        }
        else {
for(current = head; current != null; current = current.next) {
for(index = current.next; index != null; index = index.next) {
if(current.data == index.data) {
                        temp = index;
index.previous.next = index.next;
if(index.next != null)
index.next.previous = index.previous;
                        temp = null;
                    }
                }
            }
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
System.out.println();
    }
    public static void main(String[] args) {
RemoveDuplicate dList = new RemoveDuplicate();
dList.addNode(17);
dList.addNode(34);
dList.addNode(57);
dList.addNode(17);
dList.addNode(57);
dList.addNode(41);
dList.addNode(50);
dList.addNode(41);
System.out.println("Originals list: ");
```

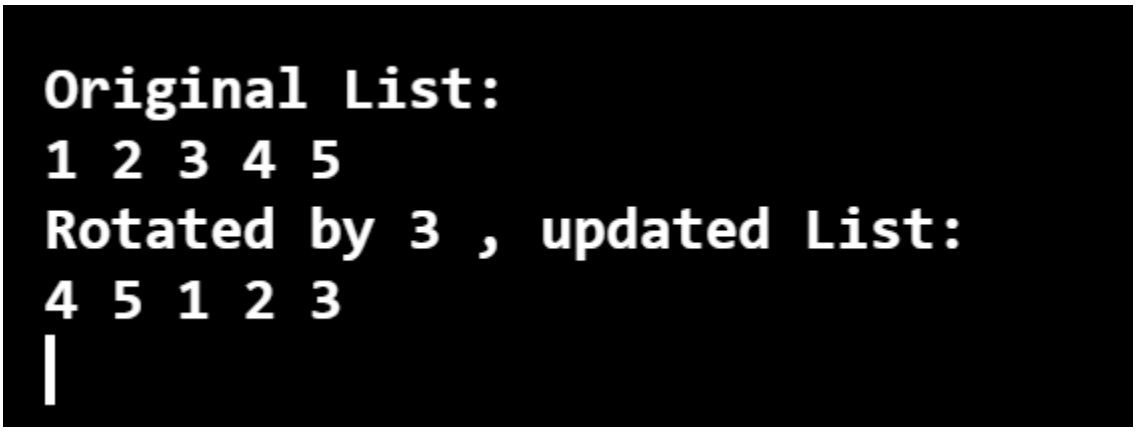
```
dList.display();
dList.removeDuplicateNode();
System.out.println("List after removing duplicates: ");
dList.display();
    }
}
```

```
Originals list:
17 34 57 17 57 41 50 41
List after removing duplicates:
17 34 57 41 50
```

14) Java program to rotate doubly linked list by N nodes.

```
public class RotateList {
    class Node{
        int data;
        Node previous;
        Node next;

        public Node(int data) {
this.data = data;
        }
    }
    int size = 0;
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
        size++;
    }
```

```
    public void rotateList(int n) {
        Node current = head;
if(n == 0 || n >= size)
            return;
        else {
for(int i = 1; i< n; i++)
                current = current.next;
tail.next = head;
            head = current.next;
head.previous = null;
            tail = current;
tail.next = null;
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
System.out.println();
    }
    public static void main(String[] args) {
RotateList dList = new RotateList();
dList.addNode(1);
dList.addNode(2);
dList.addNode(3);
dList.addNode(4);
dList.addNode(5);
System.out.println("Original List: ");
dList.display();
dList.rotateList(3);
System.out.println("Rotated by 3 , updated List: ");
dList.display();
    }
}
```



15) Java program to search an element in a doubly linked list.

```
public class SearchList {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public void searchNode(int value) {
        int i = 1;
boolean flag = false;
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
```

```
if(current.data == value) {
                flag = true;
                break;
            }
            current = current.next;
i++;
        }
        if(flag)
System.out.println("Node is present in the list at the position : " + i);
        else
System.out.println("Node is not present in the list");
    }
    public static void main(String[] args) {
SearchList dList = new SearchList();
dList.addNode(1);
dList.addNode(3);
dList.addNode(5);
dList.addNode(7);
dList.addNode(9);
dList.searchNode(4);
dList.searchNode(9);
    }
}
```

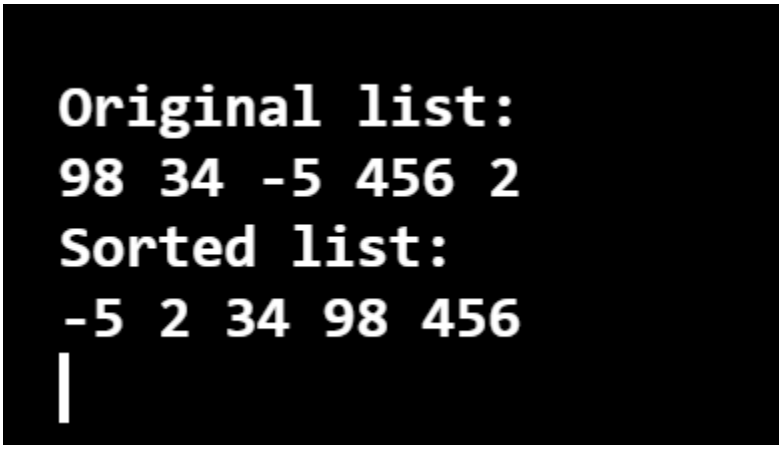
```
Node is not present in the list
Node is present in the list at the position : 5
```

16) Java program to sort the elements of the doubly linked list.

```
public class SortList {
    class Node{
        int data;
        Node previous;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    Node head, tail = null;
    public void addNode(int data) {
        Node newNode = new Node(data);
if(head == null) {
            head = tail = newNode;
head.previous = null;
tail.next = null;
        }
```

```
        else {
tail.next = newNode;
newNode.previous = tail;
            tail = newNode;
tail.next = null;
        }
    }
    public void sortList() {
        Node current = null, index = null;
        int temp;
if(head == null) {
            return;
        }
        else {
for(current = head; current.next != null; current = current.next) {
for(index = current.next; index != null; index = index.next) {
if(current.data>index.data) {
                        temp = current.data;
current.data = index.data;
index.data = temp;
                    }
                }
            }
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
            return;
        }
while(current != null) {
System.out.print(current.data + " ");
            current = current.next;
        }
System.out.println();
    }
    public static void main(String[] args) {
SortList dList = new SortList();
dList.addNode(98);
dList.addNode(34);
dList.addNode(-5);
dList.addNode(456);
dList.addNode(2);
System.out.println("Original list: ");
dList.display();
dList.sortList();
System.out.println("Sorted list: ");
```

```
dList.display();
    }
}
```



Java Circular Linked List Programs

1) Java program to create and display a Circular Linked List

```
public class CreateCircular {
    public class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    public Node head = null;
    public Node tail = null;
    public void add(int data){
        Node newNode = new Node(data);
if(head == null) {
            head = newNode;
            tail = newNode;
newNode.next = head;
        }
        else {
tail.next = newNode;
            tail = newNode;
tail.next = head;
        }
    }
    public void display() {
```

```
        Node current = head;
if(head == null) {
System.out.println("List is empty");
        }
        else {
System.out.println("Nodes of the circular linked list: ");
do{
System.out.print(" "+ current.data+" ");
                current = current.next;
}while(current != head);
System.out.println();
        }
    }
    public static void main(String[] args) {
CreateCircular cl = new CreateCircular();
cl.add(3);
cl.add(6);
cl.add(9);
cl.add(12);
cl.display();
    }
}
```

```
Nodes of the circular linked list:
 3  6  9  12
```

2) Java program to create a Circular Linked List of N nodes and count the number of nodes.

```
public class CountCircularNodes {
        public class Node{
            int data;
            Node next;
            public Node(int data) {
this.data = data;
            }
        }
        public int count;
        public Node head = null;
        public Node tail = null;
        public void add(int data){
            Node newNode = new Node(data);
```

```
if(head == null) {
            head = newNode;
            tail = newNode;
newNode.next = head;
        }
        else {
tail.next = newNode;
            tail = newNode;
tail.next = head;
        }
    }
    public void countNodes() {
        Node current = head;
do{
            count++;
            current = current.next;
}while(current != head);
System.out.println("Count of nodes present in circular linked list: "+count);
    }
    public static void main(String[] args) {
CountCircularNodes cl = new CountCircularNodes();
cl.add(4);
cl.add(8);
cl.add(12);
cl.add(16);
cl.add(20);
cl.add(24);
cl.countNodes();
    }
}
```

```
Count of nodes present in circular linked list: 6
```

3) Java program to create a Circular Linked List of n nodes and display it in reverse order.

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

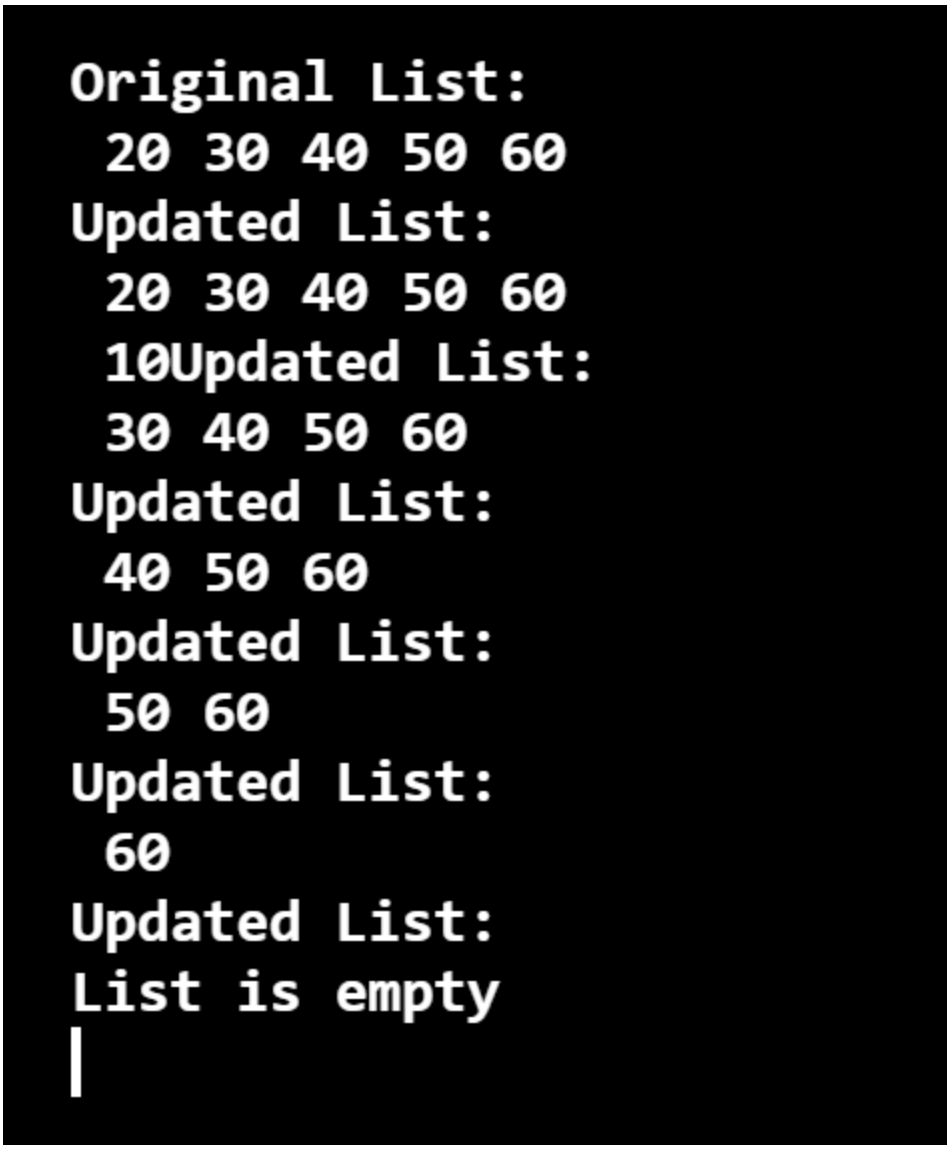
```
        public void add(int data){
            Node newNode = new Node(data);
if(head == null) {
                head = newNode;
                tail = newNode;
newNode.next = head;
            }
            else {
tail.next = newNode;
                tail = newNode;
tail.next = head;
            }
        }
        public void display() {
            Node current = head;
if(head == null) {
System.out.println("List is empty");
            }
            else {
do{
System.out.print(" "+ current.data);
                    current = current.next;
}while(current != head);
System.out.println();
            }
        }
        public void reverse(Node current) {
if(current.next == head) {
System.out.print(" "+current.data);
                return;
            }
            reverse(current.next);
System.out.print(" "+current.data);
        }
        public static void main(String[] args) {
ReverseCircularList cl = new ReverseCircularList();
cl.add(55);
cl.add(66);
cl.add(77);
cl.add(88);
cl.add(99);
cl.add(100);
System.out.println("Original List: ");
cl.display();
System.out.println("Reversed List: ");
cl.reverse(cl.head);
        }
}
```

```
Original List:
 55 66 77 88 99 100
Reversed List:
 100 99 88 77 66 55
```

) Java program to delete a node from the beginning of the Circular Linked List.

```
public class DeleteBeg {
    public class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    public Node head = null;
    public Node tail = null;
    public void add(int data){
        Node newNode = new Node(data);
if(head == null) {
            head = newNode;
            tail = newNode;
newNode.next = head;
        }
        else {
tail.next = newNode;
            tail = newNode;
tail.next = head;
        }
    }
    public void deleteStart() {
if(head == null) {
            return;
        }
        else {
if(head != tail ) {
                head = head.next;
tail.next = head;
            }
            else {
                head = tail = null;
            }
```

```
        }
    }
    public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
        }
        else {
do{
System.out.print(" "+ current.data);
                current = current.next;
}while(current != head);
System.out.println();
        }
    }
    public static void main(String[] args) {
DeleteBeg cl = new DeleteBeg();
cl.add(10);
cl.add(20);
cl.add(30);
cl.add(40);
cl.add(50);
cl.add(60);
System.out.println("Original List: ");
cl.display();
while(cl.head != null) {
cl.deleteStart();
System.out.println("Updated List: ");
cl.display();
        }
    }
}
```

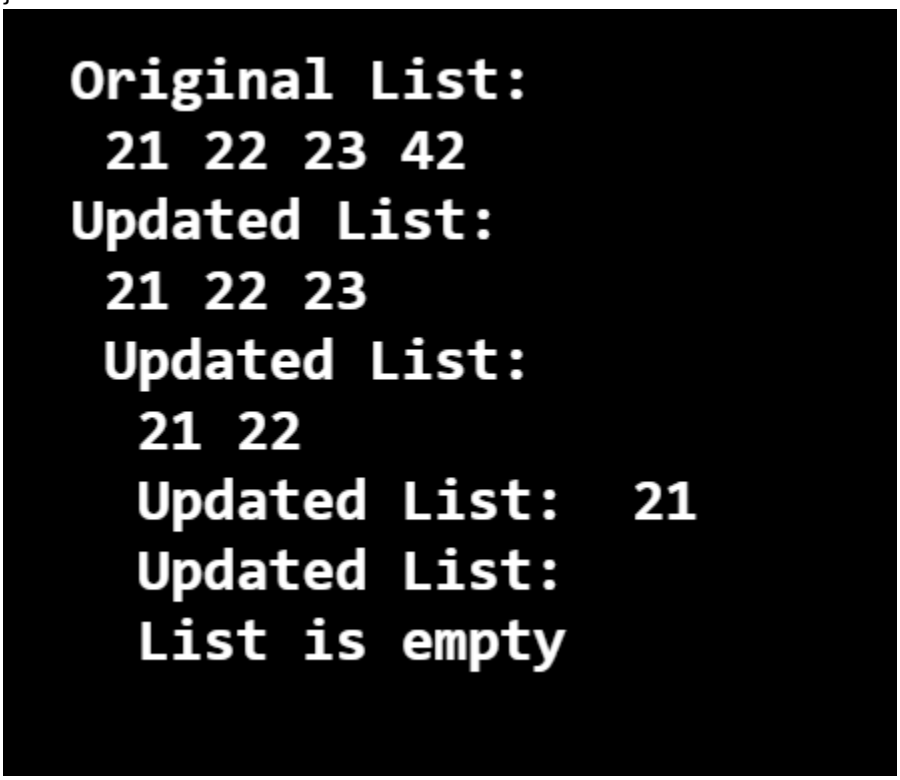


5) Java program to delete a node from the end of the Circular Linked List.

```
public class DeleteEnd {
    public class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
        }
    }
    public Node head = null;
    public Node tail = null;
    public void add(int data){
        Node newNode = new Node(data);
```

```
if(head == null) {
            head = newNode;
            tail = newNode;
newNode.next = head;
             }
             else {
tail.next = newNode;
             tail = newNode;
tail.next = head;
             }
             }
             public void deleteEnd() {
if(head == null) {
             return;
             }
             else {
if(head != tail ) {
             Node current = head;
while(current.next != tail) {
             current = current.next;
             }
             tail = current;
tail.next = head;
             }
             else {
             head = tail = null;
             }
             }
             }
             public void display() {
             Node current = head;
if(head == null) {
System.out.println("List is empty");
             }
             else {
do{
System.out.print(" "+ current.data);
             current = current.next;
}while(current != head);
System.out.println();
             }
             }
             public static void main(String[] args) {
DeleteEnd cl = new DeleteEnd();
cl.add(21);
cl.add(22);
cl.add(23);
cl.add(42);
```

```
System.out.println("Original List: ");
cl.display();
while(cl.head != null) {
cl.deleteEnd();
System.out.println("Updated List: ");
cl.display();
        }
        }
}
```

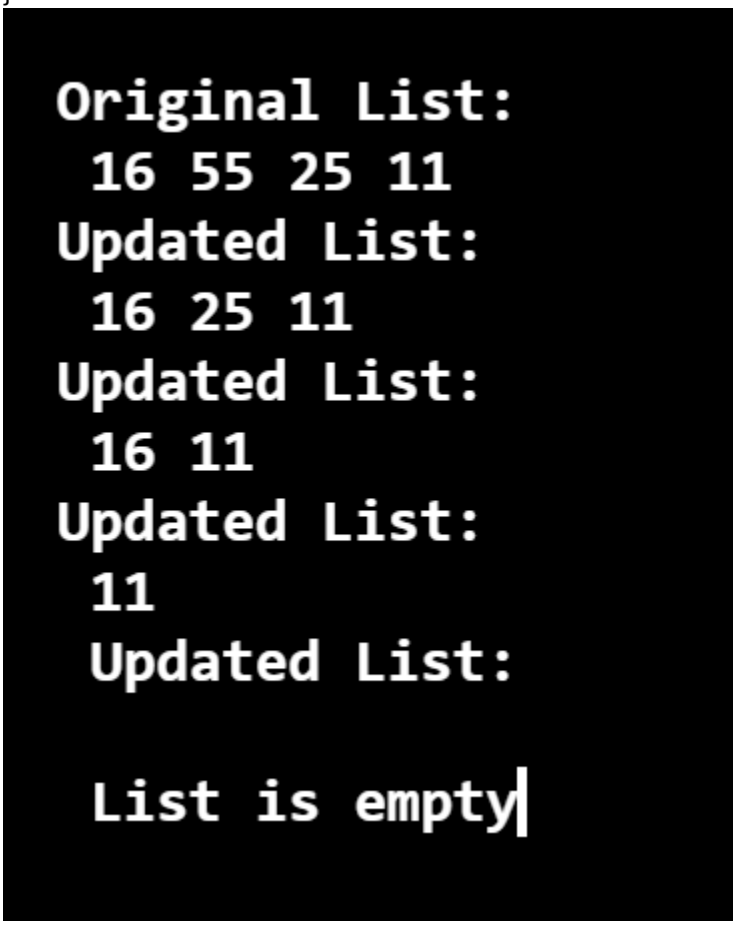


6) Java program to delete a node from the middle of the Circular Linked List.

```
public class DeleteMid {
        public class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
        }
        }
public int size;
        public Node head = null;
        public Node tail = null;
        public void add(int data){
        Node newNode = new Node(data);
if(head == null) {
```

```
            head = newNode;
            tail = newNode;
newNode.next = head;
            }
            else {
tail.next = newNode;
            tail = newNode;
tail.next = head;
            }
            size++;
            }
            public void deleteMid() {
            Node current, temp;
if(head == null) {
                return;
            }
            else {
            int count = (size % 2 == 0) ? (size/2) : ((size+1)/2);
if( head != tail ) {
            temp = head;
            current = null;
for(int i = 0; i< count-1; i++){
            current = temp;
            temp = temp.next;
            }
if(current != null) {
current.next = temp.next;
            temp = null;
            }
            else {
            head = tail = temp.next;
tail.next = head;
            temp = null;
            }
            }
            else {
            head = tail = null;
            }
            }
            size--;
            }
            public void display() {
            Node current = head;
if(head == null) {
System.out.println("List is empty");
            }
            else {
do{
```

```
System.out.print(" "+ current.data);
        current = current.next;
}while(current != head);
System.out.println();
        }
        }
        public static void main(String[] args) {
DeleteMid cl = new DeleteMid();
cl.add(16);
cl.add(55);
cl.add(25);
cl.add(11);
System.out.println("Original List: ");
cl.display();
while(cl.head != null) {
cl.deleteMid();
System.out.println("Updated List: ");
cl.display();
        }
        }
}
```



7) Java program to find the maximum and minimum value node from a circular linked list.

```
public class MinMaxInCircular {
	public class Node{
	int data;
	Node next;
	public Node(int data) {
this.data = data;
	}
	}
	public Node head = null;
	public Node tail = null;
	public void add(int data){
	Node newNode = new Node(data);
if(head == null) {
	head = newNode;
	tail = newNode;
newNode.next = head;
	}
	else {
tail.next = newNode;
	tail = newNode;
tail.next = head;
	}
	}
	public void minNode() {
	Node current = head;
	int min = head.data;
if(head == null) {
System.out.println("List is empty");
	}
	else {
do{
if(min >current.data) {
	min = current.data;
	}
	current= current.next;
}while(current != head);
System.out.println("Minimum value node in the Circular list: "+ min);
	}
	}
	public void maxNode() {
	Node current = head;
	int max = head.data;
if(head == null) {
System.out.println("List is empty");
	}
	else {
```

```
do{
if(max <current.data) {
        max = current.data;
        }
        current= current.next;
}while(current != head);

System.out.println("Maximum value node in the Circular list: "+ max);
        }
        }
        public static void main(String[] args) {
MinMaxInCircular cl = new MinMaxInCircular();
cl.add(17);
cl.add(0);
cl.add(1);
cl.add(12);
cl.minNode();
cl.maxNode();
        }
}
```

```
Minimum value node in the Circular list: 0
Maximum value node in the Circular list: 17
```

8) Java program to insert a new node at the beginning of the Circular Linked List.

```
public class InsertAtStart {
        public class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
        }
        }
        public Node head = null;
        public Node tail = null;
        public void addAtStart(int data){
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
newNode.next = head;
        }
        else {
        Node temp = head;
```

```
newNode.next = temp;
        head = newNode;
tail.next = head;
        }
        }
        public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
        }
        else {
System.out.println("Adding nodes to the start of the Circular list: ");
do{
System.out.print(" "+ current.data);
        current = current.next;
}while(current != head);
System.out.println();
        }
        }
        public static void main(String[] args) {
InsertAtStart cl = new InsertAtStart();
cl.addAtStart(8);
cl.display();
cl.addAtStart(5);
cl.display();
cl.addAtStart(4);
cl.display();
cl.addAtStart(6);
cl.display();
        }
}
```

```
Adding nodes to the start of the Circular list:
 8
Adding nodes to the start of the Circular list:
 5 8
Adding nodes to the start of the Circular list:
 4 5 8
Adding nodes to the start of the Circular list:
 6 4 5 8
```

9) Java program to insert a new node at the end of the Circular Linked List.

```
public class InsertAtEnd {
	public class Node{
	int data;
	Node next;
	public Node(int data) {
this.data = data;
	}
	}
	public Node head = null;
	public Node tail = null;
	public void addAtEnd(int data){
	Node newNode = new Node(data);
if(head == null) {
	head = newNode;
	tail = newNode;
newNode.next = head;
	}
	else {
tail.next = newNode;
	tail = newNode;
tail.next = head;
	}
	}
	public void display() {
	Node current = head;
if(head == null) {
System.out.println("List is empty");
	}
	else {
System.out.println("Adding nodes to the end of the list: ");
do{
System.out.print(" "+ current.data);
	current = current.next;
}while(current != head);
System.out.println();
	}
	}
	public static void main(String[] args) {
InsertAtEnd cl = new InsertAtEnd();
cl.addAtEnd(2);
cl.display();
cl.addAtEnd(9);
cl.display();
cl.addAtEnd(8);
cl.display();
cl.addAtEnd(5);
cl.display();
```

```
        }
}
```

```
Adding nodes to the end of the list:
 2
Adding nodes to the end of the list:
 2 9
Adding nodes to the end of the list:
 2 9 8
Adding nodes to the end of the list:
 2 9 8 5
```

10) Java program to insert a new node at the middle of the Circular Linked List.

```
public class InsertInMid {
        public class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
        }
        }
        public int size;
         public Node head = null;
        public Node tail = null;
        public void add(int data){
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
newNode.next = head;
        }
        else {
tail.next = newNode;
        tail = newNode;
tail.next = head;
        }
        size++;
        }
        public void addInMid(int data){
```

```
        Node newNode = new Node(data);
if(head == null){
        head = newNode;
        tail = newNode;
newNode.next = head;
        }
else{
        Node temp,current;
        int count = (size % 2 == 0) ? (size/2) : ((size+1)/2);
        temp = head;
        current= null;
for(int i = 0; i< count; i++){
        current = temp;
        temp = temp.next;
          }
current.next = newNode;
newNode.next = temp;
        }
        size++;
        }
        public void display() {
        Node current = head;
if(head == null) {
System.out.println("List is empty");
        }
        else {
do{
System.out.print(" "+ current.data);
        current = current.next;
}while(current != head);
System.out.println();
        }
        }
        public static void main(String[] args) {
InsertInMid cl = new InsertInMid();
cl.add(2);
cl.add(8);
cl.add(14);
cl.add(20);
System.out.println("Original list: ");
cl.display();
cl.addInMid(26);
System.out.println( "Updated List: ");
cl.display();
cl.addInMid(32);
System.out.println("Updated List: ");
cl.display();
        }
```

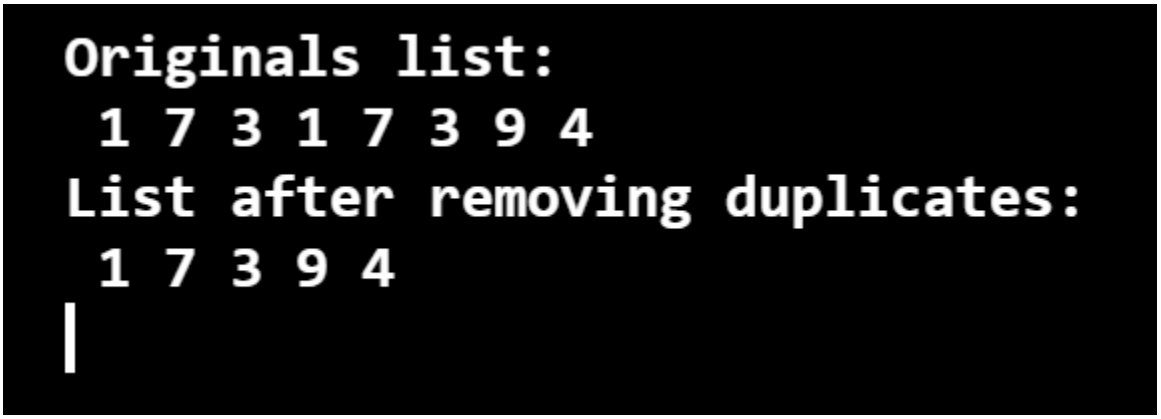
}

```
Original list:
 2 8 14 20
Updated List:
 2 8 26 14 20
 Updated List:
 2 8 26 32 14 20
```

11) Java program to remove duplicate elements from a Circular Linked List.

```
public class RemoveDuplicate {
public class Node{
int data;
Node next;
public Node(int data) {
this.data = data;
}
}
public Node head = null;
public Node tail = null;
public void add(int data){
Node newNode = new Node(data);
if(head == null) {
head = newNode;
tail = newNode;
newNode.next = head;
}
else {
tail.next = newNode;
tail = newNode;
tail.next = head;
}
}
public void removeDuplicate() {
Node current = head, index = null, temp = null;
if(head == null) {
System.out.println("List is empty");
```

```
}
else {
do{
temp = current;
index = current.next;
while(index != head) {
if(current.data == index.data) {
temp.next = index.next;
}
else {
temp = index;
}
index= index.next;
}
current =current.next;
}while(current.next != head);
}
}
public void display() {
Node current = head;
if(head == null) {
System.out.println("List is empty");
}
else {
do{
System.out.print(" "+ current.data);
current = current.next;
}while(current != head);
System.out.println();
}
}
public static void main(String[] args) {
RemoveDuplicate cl = new RemoveDuplicate();
cl.add(1);
cl.add(7);
cl.add(3);
cl.add(1);
cl.add(7);
cl.add(3);
cl.add(9);
cl.add(4);
System.out.println("Originals list: ");
cl.display();
cl.removeDuplicate();
System.out.println("List after removing duplicates: ");
cl.display();
}
}
```



12) Java program to search an element in a Circular Linked List.

```
public class SearchNode {
        public class Node{
        int data;
        Node next;
        public Node(int data) {
this.data = data;
        }
        }
        public Node head = null;
        public Node tail = null;
        public void add(int data){
        Node newNode = new Node(data);
if(head == null) {
        head = newNode;
        tail = newNode;
newNode.next = head;
        }
        else {
tail.next = newNode;
        tail = newNode;
tail.next = head;
        }
        }
        public void search(int element) {
        Node current = head;
        int i = 1;
boolean flag = false;
if(head == null) {
System.out.println("List is empty");
        }
        else {
do{
if(current.data ==  element) {
```

```
            flag = true;
            break;
            }
            current = current.next;
i++;
}while(current != head);
            if(flag)
System.out.println("Element is present in the list at the position : " + i);
            else
System.out.println("Element is not present in the list");
            }
            }
            public static void main(String[] args) {
SearchNode cl = new SearchNode();
cl.add(1);
cl.add(2);
cl.add(3);
cl.add(4);
cl.search(2);
cl.search(7);
            }
}
```

```
Element is present in the list at the position : 2
Element is not present in the list
```

13) Java program to sort the elements of the Circular Linked List.

```
public class SortList {
public class  Node{
int data;
Node next;
public Node(int data) {
this.data = data;
}
}
public Node head = null;
public Node tail = null;
public void add(int data){
Node newNode = new Node(data);
if(head == null) {
head = newNode;
tail = newNode;
newNode.next = head;
```

```
}
else {
tail.next = newNode;
tail = newNode;
tail.next = head;
}
}
public void sortList() {
Node current = head, index = null;
int temp;
if(head == null) {
System.out.println("List is empty");
}
else {
do{
index = current.next;
while(index != head) {
if(current.data>index.data) {
temp =current.data;
current.data= index.data;
index.data = temp;
}
index= index.next;
}
current =current.next;
}while(current.next != head);
}
}
public void display() {
Node current = head;
if(head == null) {
System.out.println("List is empty");
}
else {
do{
System.out.print(" "+ current.data);
current = current.next;
}while(current != head);
System.out.println();
}
}
public static void main(String[] args) {
SortList cl = new SortList();
cl.add(-7);
cl.add(9);
cl.add(-20);
cl.add(100);
cl.add(50);
```

```
System.out.println("Original list: ");
cl.display();
cl.sortList();
System.out.println("Sorted list: ");
cl.display();
}
}
```

```
Original list:
 -7 9 -20 100 50
Sorted list:
 -20 -7 9 50 100
```

Java Tree Programs

1) Java Program to calculate the Difference between the Sum of the Odd Level and the Even Level Nodes of a Binary Tree

```
import java.util.LinkedList;
import java.util.Queue;

public class DiffOddEvenLevels {
        public static class Node{
        int data;
        Node   left;
        Node right;
        public Node(int data){
this.data = data;
this.left  =  null;
this.right = null;
        }
        }
        public Node root;
        public DiffOddEvenLevels(){
        root = null;
```

```
        }
        public int difference() {
        int oddLevel = 0, evenLevel = 0, diffOddEven = 0;
      int nodesInLevel = 0;
        int currentLevel = 0;
        Queue<Node> queue = new LinkedList<Node>();
if(root == null) {
System.out.println("Tree is empty");
        return 0;
        }
        else {
queue.add(root);
currentLevel++;
        while(queue.size() != 0) {
nodesInLevel = queue.size();
while(nodesInLevel> 0) {
        Node current = queue.remove();
if(currentLevel % 2 == 0)
evenLevel += current.data;
            else
oddLevel += current.data;
if(current.left != null)
queue.add(current.left);
if(current.right != null)
queue.add(current.right);
nodesInLevel--;
        }
currentLevel++;
        }
diffOddEven = Math.abs(oddLevel - evenLevel);
        }
        return diffOddEven;
        }
        public static void main (String[] args) {
DiffOddEvenLevels bt = new DiffOddEvenLevels();
bt.root = new Node(11);
bt.root.left = new Node(22);
bt.root.right = new Node(33);
bt.root.left.left = new Node(44);
bt.root.right.left = new Node(55);
bt.root.right.right = new Node(66);
System.out.println("Difference between sum of odd level and even level nodes: " + bt.difference());
}
}
```

```
Difference between sum of odd level and even level nodes: 121
```

2) Java program to construct a Binary Search Tree and perform deletion and In-order traversal.

```
public class BinarySearchTree {
        public static class Node{
        int data;
        Node   left;
        Node right;

        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
        }
        }
        public Node root;

        public BinarySearchTree(){
        root = null;
        }
        public void insert(int data) {
        Node newNode = new Node(data);
if(root == null){
        root = newNode;
        return;
        }
        else {
        Node current = root, parent = null;

        while(true) {
        parent = current;

if(data <current.data) {
        current = current.left;
if(current == null)  {
parent.left = newNode;
                        return;
        }
        }
        else {
        current = current.right;
if(current == null)  {
parent.right = newNode;
        return;
          }
        }
        }
        }
        }
```

```
        public Node minNode(Node root) {
        if (root.left != null)
        return minNode(root.left);
        else
        return root;
        }
        public Node deleteNode(Node node, int value) {
if(node == null){
        return null;
        }
        else {
if(value <node.data)
node.left = deleteNode(node.left, value);
        else if(value >node.data)
node.right = deleteNode(node.right, value);
        else {
if(node.left == null &&node.right == null)
        node = null;
        else if(node.left == null) {
        node = node.right;
        }
        else if(node.right == null) {
        node = node.left;
        }
        else {
        Node temp = minNode(node.right);
node.data = temp.data;
node.right = deleteNode(node.right, temp.data);
        }
        }
        return node;
        }
        }
        public void inorderTraversal(Node node) {
if(root == null){
System.out.println("Tree is empty");
        return;
        }
        else {

if(node.left!= null)
inorderTraversal(node.left);
System.out.print(node.data + " ");
if(node.right!= null)
inorderTraversal(node.right);

        }
        }
```

```
	public static void main(String[] args) {

BinarySearchTree bt = new BinarySearchTree();
bt.insert(20);
bt.insert(10);
bt.insert(65);
bt.insert(50);
bt.insert(5);
bt.insert(45);
System.out.println("Binary search tree after insertion:");
bt.inorderTraversal(bt.root);
	Node deletedNode = null;
deletedNode = bt.deleteNode(bt.root, 45);
System.out.println("\nBinary search tree after deleting node 90:");
bt.inorderTraversal(bt.root);
deletedNode = bt.deleteNode(bt.root, 5);
System.out.println("\nBinary search tree after deleting node 30:");
bt.inorderTraversal(bt.root);
deletedNode = bt.deleteNode(bt.root, 10);
System.out.println("\nBinary search tree after deleting node 50:");
bt.inorderTraversal(bt.root);
	}
}
```

```
Binary search tree after insertion:
5 10 20 45 50 65
Binary search tree after deleting node 90:
5 10 20 50 65
Binary search tree after deleting node 30:
10 20 50 65
Binary search tree after deleting node 50:
20 50 65
```

3) Java program to convert Binary Tree to Binary Search Tree.

```
import java.util.Arrays;

public class ConvertBTtoBST {
	public static class Node{
	int data;
	Node left;
	Node right;
```

```
        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
        }
        }
        public Node root;

int[] treeArray;
        int index = 0;

        public ConvertBTtoBST(){
        root = null;
        }
        public Node convertBTBST(Node node) {
        int treeSize = calculateSize(node);
treeArray = new int[treeSize];
convertBTtoArray(node);
Arrays.sort(treeArray);
        Node d = createBST(0, treeArray.length -1);
        return d;
        }
        public int calculateSize(Node node)
        {
        int size = 0;
        if (node == null)
      return 0;
        else {
        size = calculateSize (node.left) + calculateSize (node.right) + 1;
        return size;
        }
        }
        public void convertBTtoArray(Node node) {
if(root == null){
System.out.println("Tree is empty");
        return;
        }
        else {
if(node.left != null)
convertBTtoArray(node.left);
treeArray[index] = node.data;
         index++;
if(node.right != null)
convertBTtoArray(node.right);
        }
        }
        public Node createBST(int start, int end) {
        if (start > end) {
```

```
        return null;
        }
        int mid = (start + end) / 2;
        Node node = new Node(treeArray[mid]);
node.left = createBST(start, mid - 1);
node.right = createBST(mid + 1, end);

        return node;
        }
        public void inorderTraversal(Node node) {
if(root == null){
System.out.println("Tree is empty");
        return;
        }
        else {

if(node.left!= null)
inorderTraversal(node.left);
System.out.print(node.data + " ");
if(node.right!= null)
inorderTraversal(node.right);

        }
        }

        public static void main(String[] args) {

ConvertBTtoBST bt = new ConvertBTtoBST();
bt.root = new Node(1);
bt.root.left = new Node(2);
bt.root.right = new Node(3);
bt.root.left.left = new Node(4);
bt.root.left.right = new Node(5);
bt.root.right.left = new Node(6);
bt.root.right.right = new Node(7);
System.out.println("Inorder representation of binary tree: ");
bt.inorderTraversal(bt.root);
        Node bst = bt.convertBTBST(bt.root);
System.out.println("\nInorder representation of resulting binary search tree: ");
bt.inorderTraversal(bst);
        }
}
```

```
Inorder representation of binary tree:
4 2 5 1 6 3 7
Inorder representation of resulting binary search tree:
1 2 3 4 5 6 7
```

4) Java program to determine whether all leaves are at same level.

```
class TreeNode
{
publicint data;
public TreeNode left;
public TreeNode right;
public TreeNode(int data)
{
this.data = data;
this.left = null;
this.right = null;
    }
}
publicclass BinaryTree
{
public TreeNode root;
publicint  depth;
public BinaryTree()
{
this.root = null;
this.depth = -1;
   }
public void checkLeafHeight(TreeNode node, int level)
{
if (node != null)
       {
if (node.left == null&& node.right == null)
           {
if (this.depth == -1)
               {
this.depth = level;
               }
elseif (this.depth != level)
               {
            this.depth = -2;
return;
               }
           }
if (this.depth != -2)
           {
                        checkLeafHeight(node.left, level + 1);
              checkLeafHeight(node.right, level + 1);
           }
       }
   }
```

```
public void isSameLevelLeaf()
{
this.depth = -1;
this.checkLeafHeight(this.root, 1);
if (this.depth <0)
        {

            System.out.println("No");
        }
else
        {

            System.out.println("Yes");
        }
    }
public static void main(String[] args)
{
// Create new tree
        BinaryTree tree = new BinaryTree();
                    tree.root = new TreeNode(1);
        tree.root.left = new TreeNode(2);
        tree.root.right = new TreeNode(3);
        tree.root.right.right = new TreeNode(6);
        tree.root.right.left = new TreeNode(5);
        tree.root.left.left = new TreeNode(4);
        tree.isSameLevelLeaf();
              tree.root.left.left.left = new TreeNode(7);
           tree.isSameLevelLeaf();
    }
}
```

```
Yes
No
```

5) Java program to determine whether two trees are identical.

```
class Node
{
    int key;
    Node left = null, right = null;

    Node(int key) {
        this.key = key;

        }
    }
```

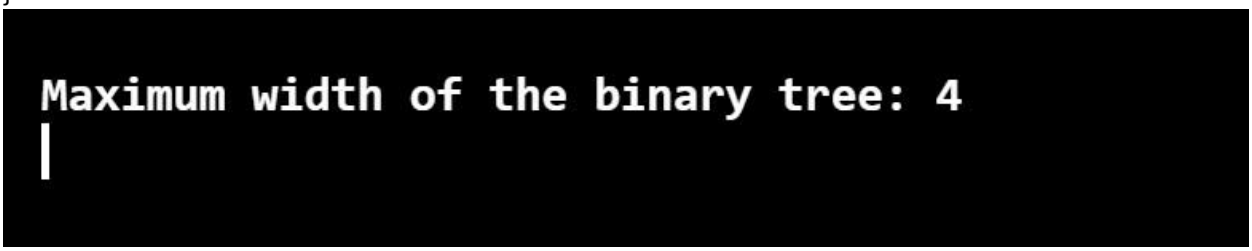
```
public class Main
{

    public static boolean isIdentical(Node x, Node y)
    {
        if (x == null && y == null) {
            return true;
        }

        return (x != null && y != null) && (x.key == y.key) &&
                    isIdentical(x.left, y.left) &&
                    isIdentical(x.right, y.right);
    }

    public static void main(String[] args)
    {

        Node x = new Node(15);
        x.left = new Node(10);
        x.right = new Node(20);
        x.left.left = new Node(8);
        x.left.right = new Node(12);
        x.right.left = new Node(16);
        x.right.right = new Node(25);


        Node y = new Node(15);
        y.left = new Node(10);
        y.right = new Node(20);
        y.left.left = new Node(8);
        y.left.right = new Node(12);
        y.right.left = new Node(16);
        y.right.right = new Node(25);

        if (isIdentical(x, y)) {
            System.out.println("The given binary trees are identical");
        }
        else {
            System.out.println("The given binary trees are not identical");
        }
    }
}
```

```
The given binary trees are identical
```

6) Java program to find maximum width of a binary tree.

```
import java.util.LinkedList;
import java.util.Queue;

public class BinaryTree {
        public static class Node{
        int data;
        Node left;
        Node right;

        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
         }
        }
        public Node root;

        public BinaryTree(){
        root = null;
        }
        public int findMaximumWidth() {
        int maxWidth = 0;
        int nodesInLevel = 0;
        Queue<Node> queue = new LinkedList<Node>();
if(root == null) {
System.out.println("Tree is empty");
        return 0;
        }
        else {
queue.add(root);
        while(queue.size() != 0) {
nodesInLevel = queue.size();
maxWidth = Math.max(maxWidth, nodesInLevel);
while(nodesInLevel> 0) {
        Node current = queue.remove();
if(current.left != null)
queue.add(current.left);
if(current.right != null)
queue.add(current.right);
nodesInLevel--;
        }
        }
        }
        return maxWidth;
        }
```

```
        public static void main(String[] args) {

BinaryTree bt = new BinaryTree();
bt.root = new Node(1);
bt.root.left = new Node(2);
bt.root.right = new Node(3);
bt.root.left.left = new Node(4);
bt.root.left.right = new Node(5);
bt.root.right.left = new Node(6);
bt.root.right.right = new Node(7);
bt.root.left.left.left = new Node(8);

System.out.println("Maximum width of the binary tree: " + bt.findMaximumWidth());
        }
}
```



7) Java program to find the largest element in a Binary Tree.

```
public class LargestNode {
        public static class Node{
        int data;
        Node left;
        Node right;

        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
        }
        }
        public Node root;

        public LargestNode(){
        root = null;
        }
        public int largestElement(Node temp){
if(root == null) {
System.out.println("Tree is empty");
        return 0;
        }
else{
```

```
        int leftMax, rightMax;
        int max = temp.data;
if(temp.left != null){
leftMax = largestElement(temp.left);
        max = Math.max(max, leftMax);
        }
if(temp.right != null){
rightMax = largestElement(temp.right);
        max = Math.max(max, rightMax);
        }
        return max;
        }
        }

        public static void main(String[] args) {

LargestNode bt = new LargestNode();
bt.root = new Node(16);
bt.root.left = new Node(22);
bt.root.right = new Node(35);
bt.root.left.left = new Node(73);
bt.root.right.left = new Node(52);
bt.root.right.right = new Node(6);
System.out.println("Largest element in the binary tree: " + bt.largestElement(bt.root));
         }
}
```

```
Largest element in the binary tree: 73
```

8) Java program to find the maximum depth or height of a tree.

```
public class BinaryTree {
        public static class Node{
        int data;
        Node left;
        Node right;

        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
        }
        }
        public Node root;
        public BinaryTree(){
        root = null;
```

```
        }
        public int findHeight(Node temp){
if(root == null) {
System.out.println("Tree is empty");
        return 0;
        }
        else {
        int leftHeight = 0, rightHeight = 0;
if(temp.left != null)
leftHeight = findHeight(temp.left);
if(temp.right != null)
rightHeight = findHeight(temp.right);
        int max = (leftHeight>rightHeight) ?leftHeight : rightHeight;
        return (max + 1);
        }
        }

        public static void main(String[] args) {

BinaryTree bt = new BinaryTree();
bt.root = new Node(0);
bt.root.left = new Node(2);
bt.root.right = new Node(3);
bt.root.left.left = new Node(4);
bt.root.left.left = new Node(3);
bt.root.right.left = new Node(5);
bt.root.right.right = new Node(6);
bt.root.right.right.right= new Node(7);
bt.root.right.right.right.right = new Node(8);
bt.root.right.right.right.right = new Node(9);
System.out.println("Maximum height of given binary tree: " + bt.findHeight(bt.root));
  }
}
```

```
Maximum height of given binary tree: 5
```

9) Java program to find the nodes which are at the maximum distance in a Binary Tree.

```
import java.util.ArrayList;

public class MaxDistance {
        public static class Node{
        int data;
        Node left;
```

```
        Node right;

        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
        }
        }
        public Node root;

int[] treeArray;
        int index = 0;

        public MaxDistance(){
        root = null;
        }
        public int calculateSize(Node node)
        {
        int size = 0;
        if (node == null)
        return 0;
        else {
        size = calculateSize (node.left) + calculateSize (node.right) + 1;
        return size;
        }
        }
        public void convertBTtoArray(Node node) {
if(root == null){
System.out.println("Tree is empty");
        return;
        }
        else {
if(node.left != null)
convertBTtoArray(node.left);
treeArray[index] = node.data;
        index++;
if(node.right != null)
convertBTtoArray(node.right);
        }
        }

        public int getDistance(Node temp, int n1) {
        if (temp != null) {
            int x = 0;
        if ((temp.data == n1) || (x = getDistance(temp.left, n1)) > 0
        || (x = getDistance(temp.right, n1)) > 0) {

        return x + 1;
```

```
        }
        return 0;
        }
        return 0;
        }
        public Node lowestCommonAncestor(Node temp, int node1, int node2) {
        if (temp != null) {
        if (temp.data == node1 || temp.data == node2) {
        return temp;
        }
        Node left = lowestCommonAncestor(temp.left, node1, node2);
        Node right = lowestCommonAncestor(temp.right, node1, node2);

        if (left != null && right != null) {
        return temp;
        }

            if (left != null) {
        return left;
        }
        if (right != null) {
        return right;
        }
        }
        return null;
        }
        public int findDistance(int node1, int node2) {
    int d1 = getDistance(root, node1) - 1;
        int d2 = getDistance(root, node2) - 1;
        Node ancestor = lowestCommonAncestor(root, node1, node2);

        int d3 = getDistance(root, ancestor.data) - 1;
        return (d1 + d2) - 2 * d3;
        }

        public void nodesAtMaxDistance(Node node) {
        int maxDistance = 0, distance = 0;
ArrayList<Integer>arr = new ArrayList<>();
        int treeSize = calculateSize(node);
treeArray = new int[treeSize];
convertBTtoArray(node);
for(int i = 0; i<treeArray.length; i++) {
for(int j = i; j <treeArray.length; j++) {
        distance = findDistance(treeArray[i], treeArray[j]);
if(distance >maxDistance) {
maxDistance = distance;
arr.clear();
arr.add(treeArray[i]);
```

```
arr.add(treeArray[j]);
            }
            else if(distance == maxDistance) {
arr.add(treeArray[i]);
arr.add(treeArray[j]);
            }
            }
            }
System.out.println("Nodes which are at maximum distance: ");
for(int i = 0; i<arr.size(); i = i + 2) {
System.out.println("( " + arr.get(i) + "," + arr.get(i+1) + " )");
            }
            }

            public static void main(String[] args) {

MaxDistance bt = new MaxDistance();
bt.root = new Node(1);
bt.root.left = new Node(2);
bt.root.right = new Node(3);
bt.root.left.left = new Node(4);
bt.root.left.right = new Node(5);
bt.root.right.left = new Node(6);
bt.root.right.right = new Node(7);
bt.root.right.right.right = new Node(8);
bt.root.right.right.right.left = new Node(9);
bt.nodesAtMaxDistance(bt.root);
            }
}
```

```
Nodes which are at maximum distance:
( 4,9 )
( 5,9 )
```

10) Java program to find the smallest element in a tree.

```
public class SmallestNode {
    public static class Node{
            int data;
            Node left;
            Node right;

            public Node(int data){
this.data = data;
```

```
this.left = null;
this.right = null;
        }
        }

        public Node root;

        public SmallestNode(){
        root = null;
        }
        public int smallestElement(Node temp){
if(root == null) {
System.out.println("Tree is empty");
        return 0;
        }
        else {
        int leftMin, rightMin;
        int min = temp.data;

if(temp.left != null){
leftMin = smallestElement(temp.left);
        min = Math.min(min, leftMin);
        }
if(temp.right != null){
rightMin = smallestElement(temp.right);
        min = Math.min(min, rightMin);
        }
        return min;
        }
        }

        public static void main(String[] args) {

SmallestNode bt = new SmallestNode();
bt.root = new Node(2);
bt.root.left = new Node(0);
bt.root.right = new Node(-1);
bt.root.left.left = new Node(1);
bt.root.right.left = new Node(8);
bt.root.right.right = new Node(5);

System.out.println("Smallest element in the binary tree: " + bt.smallestElement(bt.root));
        }
}
```

```
Smallest element in the binary tree: -1
```

11) Java program to find the sum of all the nodes of a binary tree.

```
public class SumOfNodes {

        public static class Node{
        int data;
        Node left;
        Node right;

        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
        }
        }
        public Node root;

        public SumOfNodes(){
        root = null;
        }

        public int calculateSum(Node temp){
        int sum, sumLeft, sumRight;
        sum = sumRight = sumLeft = 0;
if(root == null) {
System.out.println("Tree is empty");
        return 0;
        }
        else {
if(temp.left != null)
sumLeft = calculateSum(temp.left);
if(temp.right != null)
sumRight = calculateSum(temp.right);
        sum = temp.data + sumLeft + sumRight;
           return sum;
        }
        }

        public static void main(String[] args) {
```

```
SumOfNodes bt = new SumOfNodes();
bt.root = new Node(7);
bt.root.left = new Node(12);
bt.root.right = new Node(3);
bt.root.left.left = new Node(1);
bt.root.right.left = new Node(0);
bt.root.right.right = new Node(2);
System.out.println("Sum of all nodes of binary tree: " + bt.calculateSum(bt.root));
        }
        }
```

```
Sum of all nodes of binary tree: 25
```

12) Java program to find the total number of possible Binary Search Trees with N keys.

```
public class BinarySearchTree {

        public static class Node{
        int data;
        Node left;
        Node right;

        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
        }
        }

        public Node root;

        public BinarySearchTree(){
        root = null;
        }
        public int factorial(int num) {
        int fact = 1;
if(num == 0)
        return 1;
        else {
while(num> 1) {
        fact = fact * num;
num--;
        }
        return fact;
```

```
        }
        }

        public int numOfBST(int key) {
        int catalanNumber = factorial(2 * key)/(factorial(key + 1) * factorial(key));
        return catalanNumber;
        }

        public static void main(String[] args) {

BinarySearchTree bt = new BinarySearchTree();

System.out.println("Total number of possible Binary Search Trees with given key: " + bt.numOfBST(4));
        }
}
```

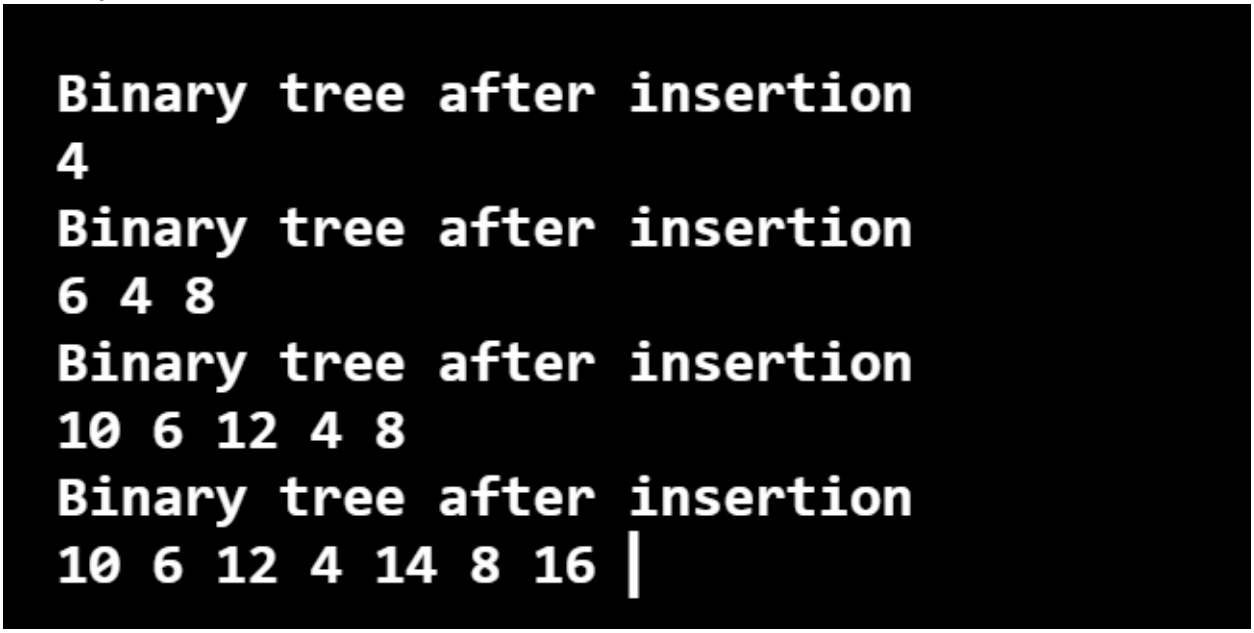
```
Total number of possible Binary Search Trees with given key: 14
```

13) Java program to implement Binary Tree using the Linked List.

```
import java.util.LinkedList;
import java.util.Queue;

public class BinaryTree {

        public static class Node{
        int data;
        Node left;
        Node right;

        public Node(int data){
this.data = data;
this.left = null;
this.right = null;
        }
        }
        public Node root;

        public BinaryTree(){
        root = null;
        }
        public void insertNode(int data) {
        Node newNode = new Node(data);

if(root == null){
        root = newNode;
```

```
        return;
        }
        else {
        Queue<Node> queue = new LinkedList<Node>();
queue.add(root);

        while(true) {

        Node node = queue.remove();
if(node.left != null &&node.right != null) {
queue.add(node.left);
queue.add(node.right);
        }
        else {
if(node.left == null) {
node.left = newNode;
queue.add(node.left);
        }
        else {
node.right = newNode;
queue.add(node.right);
        }
        break;
        }
        }
        }
        }
        public void inorderTraversal(Node node) {

if(root == null){
System.out.println("Tree is empty");
        return;
        }
        else {

if(node.left!= null)
inorderTraversal(node.left);
System.out.print(node.data + " ");
if(node.right!= null)
inorderTraversal(node.right);

        }
        }

        public static void main(String[] args) {

BinaryTree bt = new BinaryTree();
```

```
bt.insertNode(4);
System.out.println("Binary tree after insertion");
bt.inorderTraversal(bt.root);

bt.insertNode(6);
bt.insertNode(8);
System.out.println("\nBinary tree after insertion");
bt.inorderTraversal(bt.root);

bt.insertNode(10);
bt.insertNode(12);
System.out.println("\nBinary tree after insertion");
bt.inorderTraversal(bt.root);

bt.insertNode(14);
bt.insertNode(16);
System.out.println("\nBinary tree after insertion");
bt.inorderTraversal(bt.root);

        }
        }
```



14) Java program to search a node in a Binary Tree.

```
public class SearchBinaryTree {
        public static class Node{
        int data;
        Node   left;
        Node right;
```

```
	public Node(int data){
this.data = data;
this.left = null;
this.right = null;
	}
	}
	public Node root;

	public static boolean flag = false;

	public SearchBinaryTree(){
	root = null;
	}

	public void searchNode(Node temp, int value){
if(root == null){
System.out.println("Tree is empty");
	}
else{
if(temp.data == value){
	flag = true;
	return;
	}
if(flag == false &&temp.left != null){
searchNode(temp.left, value);
	}
if(flag == false &&temp.right != null){
searchNode(temp.right, value);
	}
	}
	}

	public static void main(String[] args) {

SearchBinaryTree bt = new SearchBinaryTree();
bt.root = new Node(23);
bt.root.left = new Node(11);
bt.root.right = new Node(45);
bt.root.left.left = new Node(99);
bt.root.right.left = new Node(52);
bt.root.right.right = new Node(60);
bt.searchNode(bt.root, 99);

	if(flag)
System.out.println("Element is present in the binary tree");
	else
System.out.println("Element is not present in the binary tree");
```

```
Element is present in the binary tree
```