**Practical No. 1:** Write a program to insert a string into another string (Without using any predefined method) at any given index.

**Source Code:**

import java.util.Scanner;

class StringAppend

{

    String insertString(String str , int index , String append)

    {

        char[] newStr = new char[str.length() + append.length()];

        int i = 0;

        for(i = 0 ; i < index ; i++)

        {

            newStr[i] = str.charAt(i);

        }

        for(int j = 0 ; j < append.length(); i++,j++)

        {

            newStr[i] = append.charAt(j);

        }

        for(int k = index ; k < str.length() ; i++,k++)

        {

            newStr[i] = str.charAt(k);

        }

        return new String(newStr);

    }

}

public class Q1 {

    public static void main(String args[])

    {

        StringAppend obj = new StringAppend();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter String:");

        String str = sc.nextLine();

        System.out.print("Enter Index: ");

        int ind = sc.nextInt();

        System.out.print("Enter appended String:");

        String appendStr = sc.next();

        String newStr = obj.insertString(str, ind, appendStr);

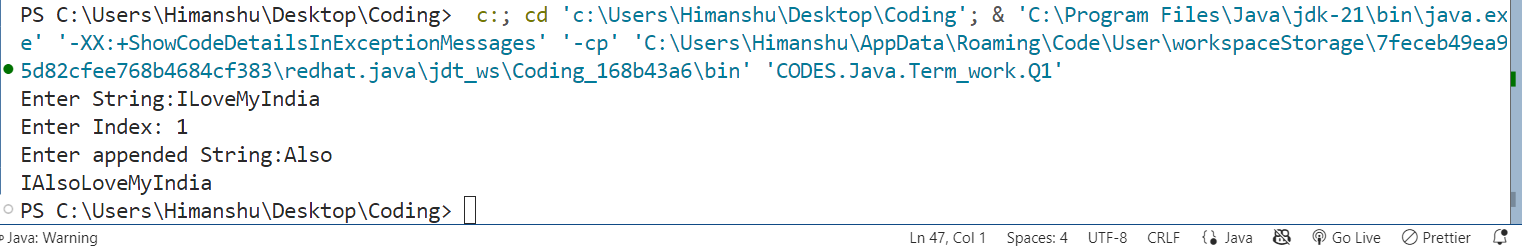
        System.out.println(newStr);

        sc.close();

    }

}

**Output:**



**Practical No. 2:** Write a program to check two strings are Anagram of each other.

**Source Code:**

import java.util.Arrays;

import java.util.Scanner;

class Anagram

{

    void sortString(char arr[])

    {

        int n = arr.length;

        for(int i = 0 ; i < n ; i++)

        {

            for(int j = 0 ; j < n-i-1 ; j++)

            {

                if(arr[j] > arr[j+1])

                {

                    char temp = arr[j];

                    arr[j] = arr[j+1];

                    arr[j+1] = temp;

                }

            }

        }

    }

    boolean checkAnagram(String str1 , String str2)

    {

        if(str1.length() != str2.length())

        {

            return false;

        }

        char[] arr1 = str1.toCharArray();

        char[] arr2 = str2.toCharArray();

        sortString(arr1);

        sortString(arr2);

        return Arrays.equals(arr1,arr2);

    }

}

public class Q2 {

    public static void main(String args[])

    {

        Anagram obj = new Anagram();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter String 1: ");

        String s1 = sc.nextLine();

        System.out.print("Enter String 2: ");

        String s2 = sc.next();

        if(obj.checkAnagram(s1 , s2))

        {

            System.out.println("Both Strings are anagram of each other");

        }else

        {

            System.out.println("Both Strings are not anagram of each other");

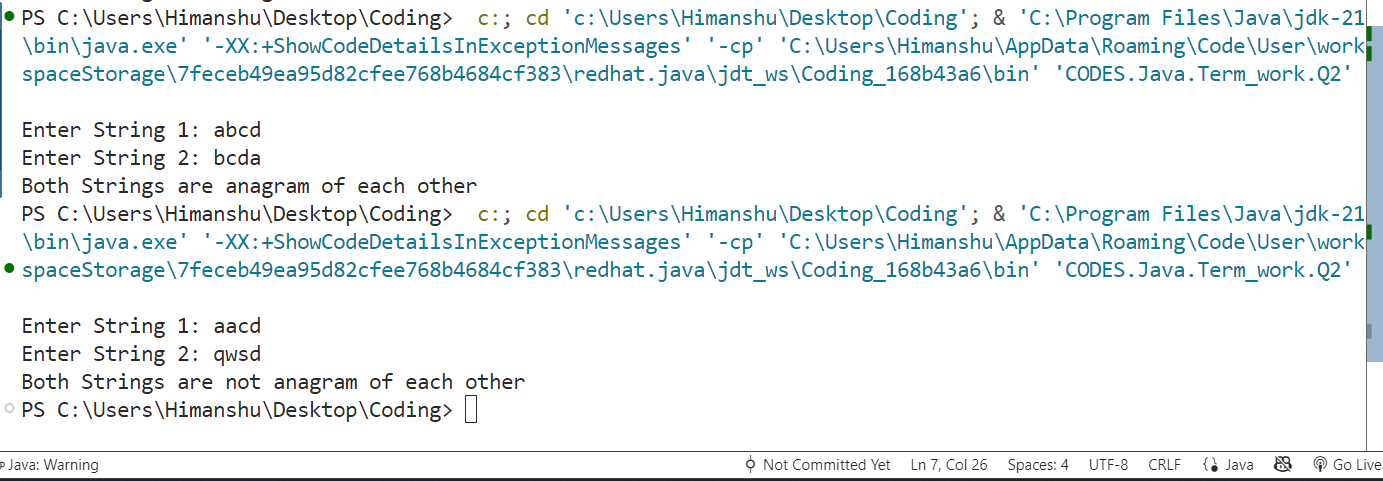
        }

        sc.close();

    }

}

**Output:**



**Practical No. 3:** Java program for Sorting a String:  
(i)Without using any inbuilt sorting functions   
(ii) By using inbuilt function

**Source Code:**

import java.util.Arrays;

import java.util.Scanner;

class StringSorting

{

    void sortingWithoutInbuiltFunction(char str[])

    {

        int n = str.length;

        for(int i = 0 ; i < n-1 ; i++)

        {

            for(int j = 0 ; j < n-i-1 ; j++)

            {

                if(str[j] > str[j+1])

                {

                    char temp = str[j];

                    str[j] = str[j+1];

                    str[j+1] = temp;

                }

            }

        }

    }

    void sortingWithInbuiltFunction(char str[])

    {

        Arrays.sort(str);

    }

}

public class Q3 {

    public static void main(String[] args)

    {

        StringSorting obj = new StringSorting();

        System.out.print("Enter string: ");

        Scanner sc = new Scanner(System.in);

        String input = sc.nextLine();

        char[] str1 = input.toCharArray();

        char[] str2 = input.toCharArray();

        // Sorting without inbuilt function

        obj.sortingWithoutInbuiltFunction(str1);

        System.out.println("Sorted without inbuilt function: " + new String(str1));

        // Sorting with inbuilt function

        obj.sortingWithInbuiltFunction(str2);

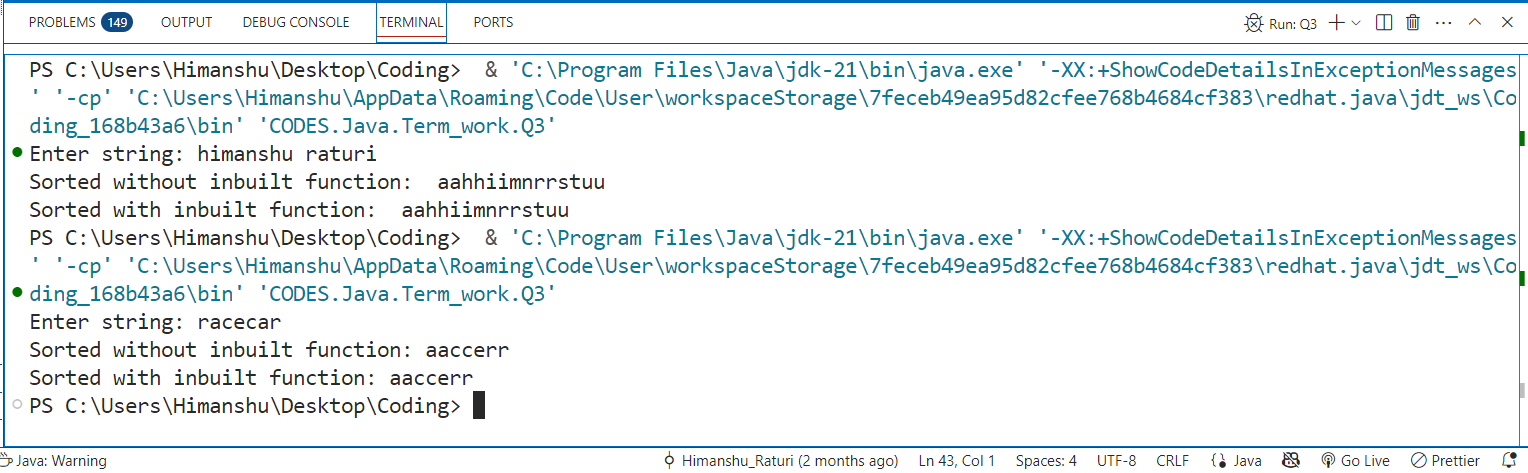
        System.out.println("Sorted with inbuilt function: " + new String(str2));

        sc.close();

    }

}

**Output:**



**Practical No. 4:** Program to Extract Substring from a String with Equal 0, 1, and 2.

**Source Code:**

import java.util.Scanner;

public class Q4 {

        public static int longestEqual012Substring(String str) {

            int maxLength = 0;

            int n = str.length();

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

                int count0 = 0, count1 = 0, count2 = 0;

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

                    char ch = str.charAt(j);

                    if (ch == '0') count0++;

                    else if (ch == '1') count1++;

                    else if (ch == '2') count2++;

                    if (count0 == count1 && count1 == count2) {

                        maxLength = Math.max(maxLength, j - i + 1);

                    }

                }

            }

            return maxLength;

        }

        public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);

            System.out.print("Enter String: ");

            String str = sc.nextLine();

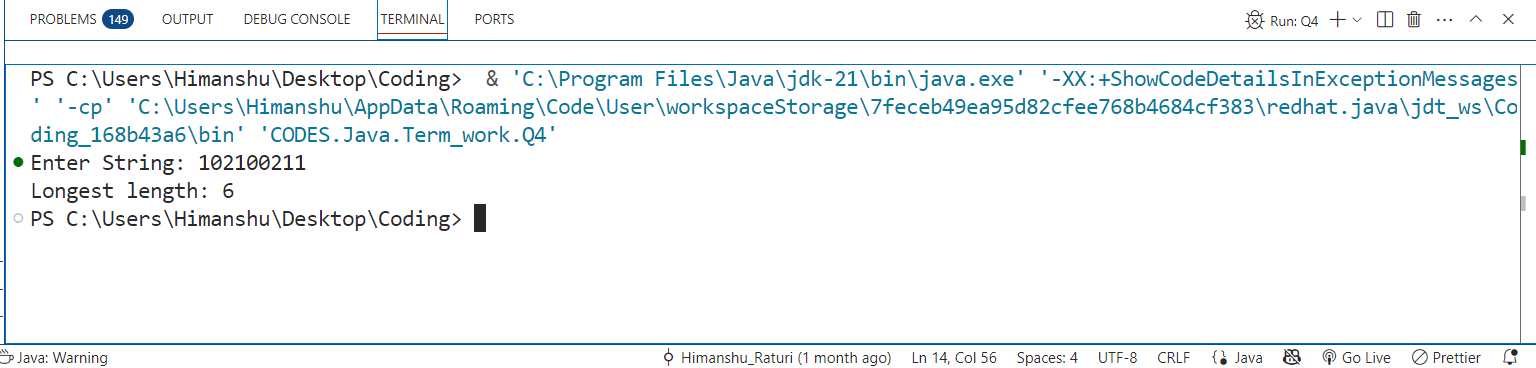
            System.out.println("Longest length: " + longestEqual012Substring(str));

            sc.close();

        }

}

**Output:**

****

**Practical No. 5:** Write a program to validate an IPv4 Address.IPv4 addresses are canonically represented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots, e.g., 172.16.254.1

**Source Code:**

import java.util.Scanner;

public class Q5 {

    public static boolean isValidIPv4(String ip) {

        String[] parts = ip.split("\\.");

        if (parts.length != 4) {

            return false;

        }

        for (String part : parts)

        {

            int num = Integer.parseInt(part);

            if (num < 0 || num > 255) {

                return false;

            }

            if (!part.equals(String.valueOf(num)))

            {

                return false;

            }

        }

        return true;

    }

    public static void main(String[] args) {

        System.out.print("Enter String: ");

        Scanner sc = new Scanner(System.in);

        String ip = sc.nextLine();

        if (isValidIPv4(ip)) {

            System.out.println("Valid");

        } else {

            System.out.println("Not Valid");

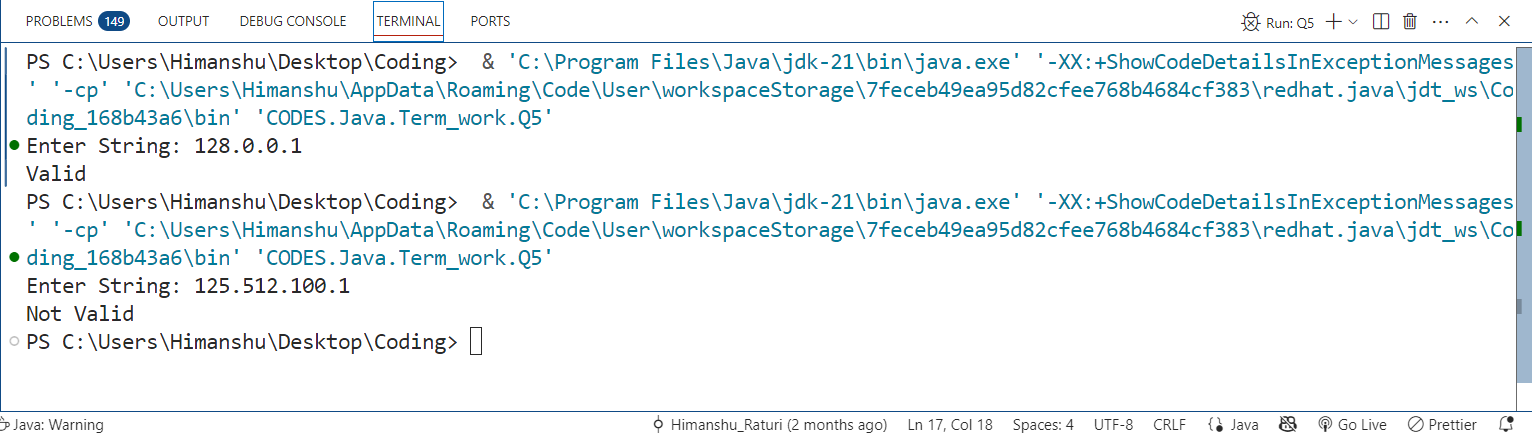
        }

        sc.close();

    }

}

**Output:**



**Practical No. 6:** Print all permutations of a string in Java and permutations need to be distinct.

**Source Code:**

import java.util.Scanner;

public class Q6 {

     static void sort(char[] arr) {

        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {

            for (int j = 0; j < n - i - 1; j++) {

                if (arr[j] > arr[j + 1]) {

                    char temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                }

            }

        }

    }

    static void generatePermutations(char[] chars, boolean[] used, char[] result, int depth) {

        if (depth == chars.length) {

            System.out.println(new String(result));

            return;

        }

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

            if (used[i]) continue;

            if (i > 0 && chars[i] == chars[i - 1] && !used[i - 1]) continue;

            used[i] = true;

            result[depth] = chars[i];

            generatePermutations(chars, used, result, depth + 1);

            used[i] = false;

        }

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter String: ");

        String str = sc.nextLine();

        char[] chars = str.toCharArray();

        sort(chars);

        boolean[] used = new boolean[chars.length];

        char[] result = new char[chars.length];

        System.out.println("Distinct permutations of \"" + str + "\":");

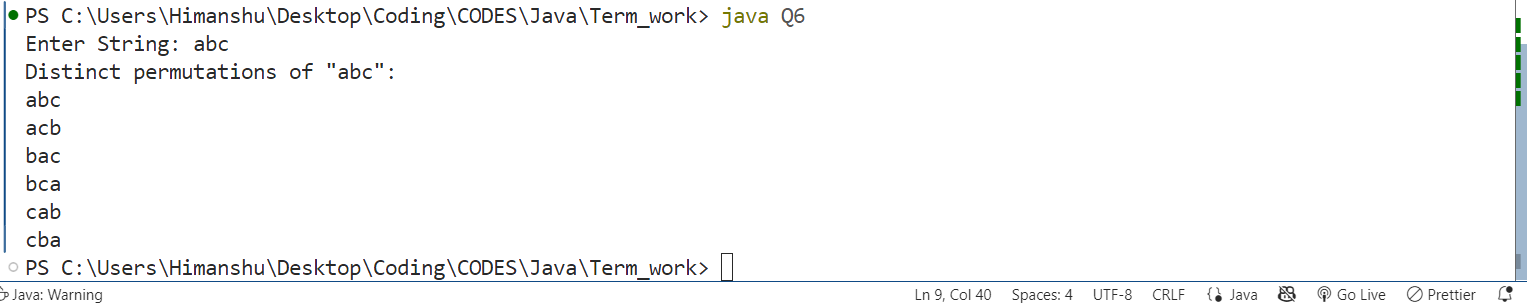
        generatePermutations(chars, used, result, 0);

        sc.close();

    }

}

**Output:**



**Practical No. 7:** Find out if there are any occurrences of the word "city" in a sentence:   
Example: Input: I love my city. My city is clean. It is a popular city.

**Source Code:**

import java.util.Scanner;

public class Q7 {

    static int countOccurance(String[] arr, String word)

    {

        int count = 0;

        for(String s : arr)

        {

            if(s.equals(word))

            count++;

        }

        return count;

    }

    public static void main(String args[])

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a String: ");

        String sent = sc.nextLine();

        System.out.print("Enter target: ");

        String t = sc.nextLine();

        sent = sent.replaceAll("[^a-zA-Z]" , " ").toLowerCase();

        String word[] = sent.split(" ");

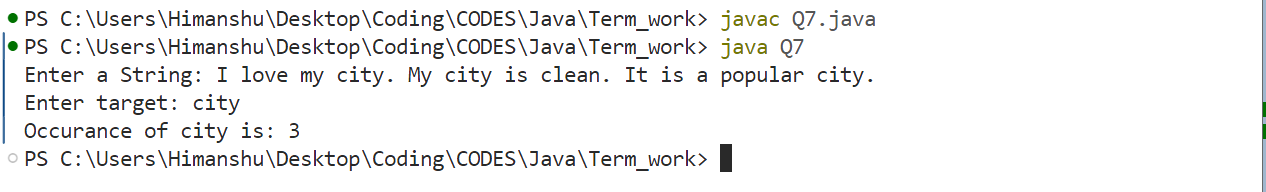
        System.out.println("Occurance of " + t + " is: " + countOccurance(word, t));

        sc.close();

    }

}

**Output:**



**Practical No. 8:** Check if Email Address is Valid or not in Java.

**Source Code:**

import java.util.regex.\*;

import java.util.Scanner;

public class Q8 {

    static boolean verifyEmail(String input)

    {

        String emailRegex = "^[A-Za-z0-9+\_.-]+@[A-Za-z0-9.-]+$";

        Pattern pattern = Pattern.compile(emailRegex);

        Matcher matcher = pattern.matcher(input);

        return matcher.matches();

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Email: ");

        String input = sc.nextLine();

        if(verifyEmail(input))

        {

            System.out.println("Valid Email.");

        }else

        {

            System.out.println("Not a valid Email.");

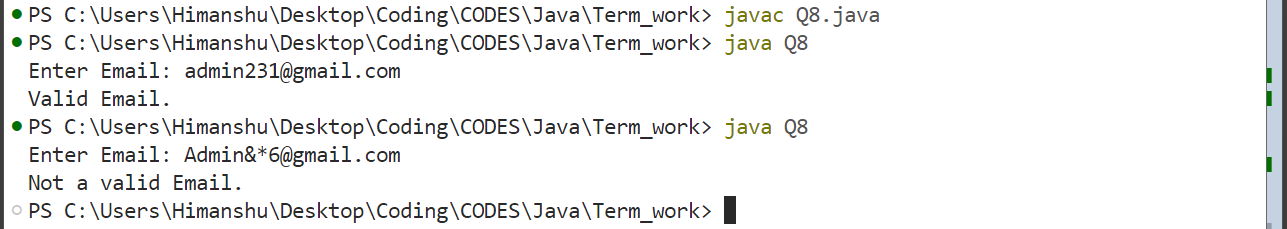
        }

        sc.close();

    }

}

**Output:**



**Practical No. 9:** We are given two arrays that represent the arrival and departure times of trains, the task is to find the minimum number of platforms required so that no train waits.

**Source Code:**  
import java.util.Arrays;

import java.util.Scanner;

public class Q9 {

    static int findMinPlatforms(int[] arrival, int[] departure) {

        int n = arrival.length;

        Arrays.sort(arrival);

        Arrays.sort(departure);

        int platforms = 1, maxPlatforms = 1;

        int i = 1, j = 0;

        while (i < n && j < n) {

            if (arrival[i] <= departure[j]) {

                platforms++;

                i++;

            } else {

                platforms--;

                j++;

            }

            maxPlatforms = Math.max(maxPlatforms, platforms);

        }

        return maxPlatforms;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int[] arrival = new int[5];

        int[] departure = new int[5];

        System.out.println("Enter Arrival Array: ");

        for(int i = 0 ; i < 5; i++)

        {

            arrival[i] = sc.nextInt();

        }

        System.out.println("Enter Departure Array: ");

        for(int i = 0 ; i < 5; i++)

        {

            departure[i] = sc.nextInt();

        }

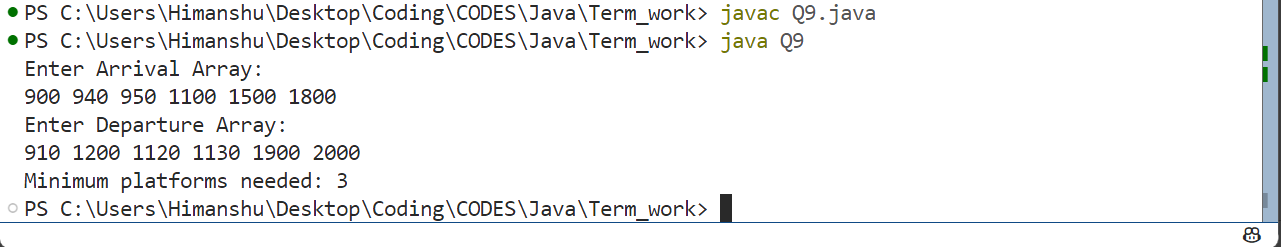
        System.out.println("Minimum platforms needed: " + findMinPlatforms(arrival, departure));

        sc.close();

    }

}

**Output:**



**Practical No. 10:** Given an unsorted array of integers, sort the array into a wave array. An array arr[0..n-1] is sorted in wave form if:   
arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= .....

**Source Code:**

import java.util.Arrays;

import java.util.Scanner;

public class Q10 {

    static void convertWave(int[] arr) {

        int n = arr.length;

        for (int i = 0; i < n - 1; i += 2) {

            if (i > 0 && arr[i] < arr[i - 1]) {

                int temp = arr[i];

                arr[i] = arr[i - 1];

                arr[i - 1] = temp;

            }

            if (i < n - 1 && arr[i] < arr[i + 1]) {

                int temp = arr[i];

                arr[i] = arr[i + 1];

                arr[i + 1] = temp;

            }

        }

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter array: ");

        int arr[] = new int[8];

        for(int i = 0 ; i < 8 ; i++)

        {

            arr[i] = sc.nextInt();

        }

        convertWave(arr);

        System.out.println("Wave Form Array:");

        for(int x: arr)

        {

            System.out.print(x);

        }

        sc.close();

    }

}

**Output:**

