**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;

public class insertstring{

public static String insert(String s1, String s2, int index){

char[] str1=s1.toCharArray();

char[] str2=s2.toCharArray();

char[] result=new char[str1.length+str2.length];

int i=0,j=0;

while(i<index){

result[j++]=str1[i++];

}

for(char c:str2){

result[j++]=c;

}

while(i<str1.length){

result[j++]=str1[i++];

}

String newstr= "";

for(char c:result){

newstr+=c;

}

return newstr;

}

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

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

String s1=sc.nextLine();

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

String s2=sc.nextLine();

System.out.println("Enter Index where to enter: ");

int index=sc.nextInt();

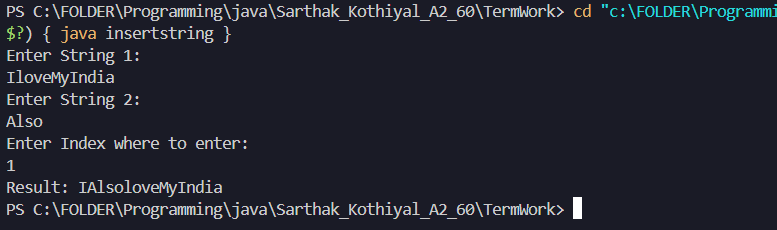
String result=insert(s1,s2,index);

System.out.println("Result: " + result);

}

}

**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;

public class anagram{

    public static boolean check(String *s1*, String *s2*) {

        s1 = s1.replaceAll("\\s", "").toLowerCase();

        s2 = s2.replaceAll("\\s", "").toLowerCase();

        if (s1.length() != s2.length()) {

            return false;

        }

        char[] a1 = s1.toCharArray();

        char[] a2 = s2.toCharArray();

        Arrays.sort(a1);

        Arrays.sort(a2);

        return Arrays.equals(a1, a2);

    }

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

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter first string:");

        String str1 = sc.nextLine();

        System.out.println("Enter second string:");

        String str2 = sc.nextLine();

        if (check(str1, str2)) {

            System.out.println("The strings are anagrams.");

        }

        else {

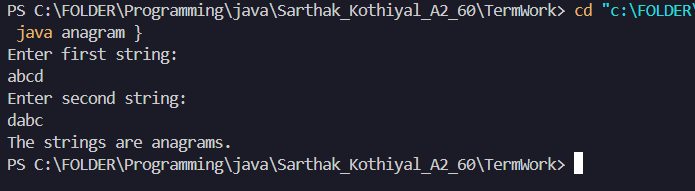
            System.out.println("The strings are NOT anagrams.");

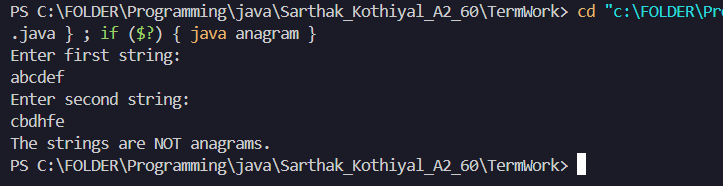
        }

    }

}

**Output:**





**Practical No. 3:**

Java program for Sorting a String

(i)Without using any inbuilt sorting functions

(ii) By using inbuilt functions

**Source Code:**

**(i)Without using any inbuilt sorting functions**

import java.util.Scanner;

public class sortwithout {

public static String sortString(String input) {

char[] chars = input.toCharArray();

for (int i = 0; i < chars.length - 1; i++) {

for (int j = 0; j < chars.length - i - 1; j++) {

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

char temp = chars[j];

chars[j] = chars[j + 1];

chars[j + 1] = temp;

}

}

}

String sorted = "";

for (char c : chars) {

sorted += c;

}

return sorted;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string:");

String input = sc.nextLine();

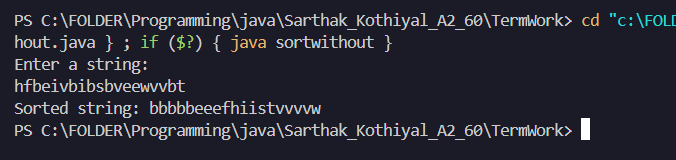
String sorted = sortString(input);

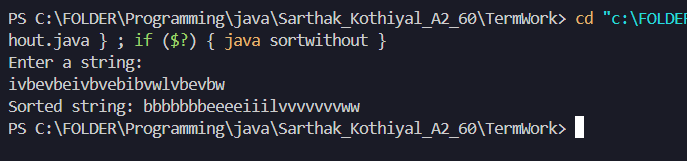
System.out.println("Sorted string: " + sorted);

}

}

**Output:**





**(ii) By using inbuilt functions**

import java.util.Arrays;

import java.util.Scanner;

public class sortwith {

    public static String sortString(String *input*) {

        char[] chars = input.toCharArray();

        Arrays.sort(chars);

        return new String(chars);

    }

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

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a string to sort (using inbuilt sort):");

        String input = sc.nextLine();

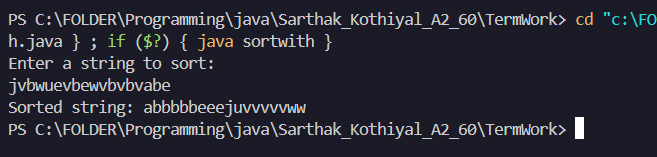
        String sorted = sortString(input);

        System.out.println("Sorted string: " + sorted);

    }

}

**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 equalsubstring {

public static void find(String str) {

int n = str.length();

boolean found = false;

System.out.println("Substring with equal number of 0, 1, and 2:");

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 && count0 != 0) {

found = true;

System.out.println(str.substring(i, j + 1));

}

}

}

if (!found) {

System.out.println("No substring exist.");

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string:");

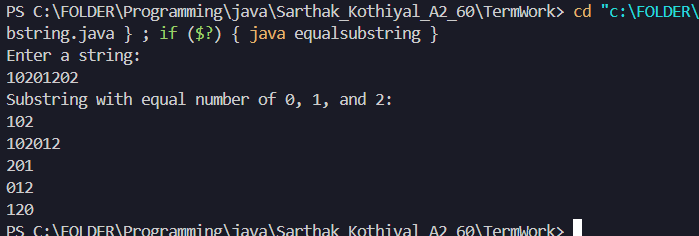
String input = sc.nextLine();

find(input);

}

}

**Output:**

****

**Practical No. 5:**

Write a program to validate an IPv4 Address.

**Source Code:**

import java.util.Scanner;

public class ipv4 {

public static boolean check(String ip) {

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

if (parts.length != 4){

return false;

}

for (String part : parts) {

if (part.length() > 1 && part.startsWith("0")) {

return false;

}

int num = Integer.parseInt(part);

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

return false;

}

}

return true;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter an IPv4 address:");

String ip = sc.nextLine();

if(check(ip)) {

System.out.println("Valid IPv4 address.");

}

else {

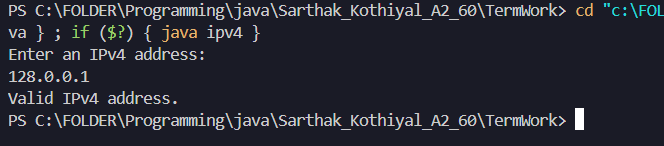
System.out.println("Invalid IPv4 address.");

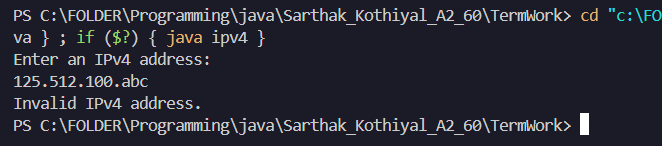
}

}

}

**Output:**

****

****

**Practical No. 6:**

Print all permutations of a string in Java and permutations need to be distinct.

**Source Code:**

import java.util.\*;

public class permutation{

    static void swap(StringBuffer *str*,int *a*,int *b*){

        char ch=str.charAt(a);

        str.setCharAt(a,str.charAt(b));

        str.setCharAt(b,ch);

    }

    static void permu(StringBuffer *str*,int *low*,int *high*){

        if(low==high){

            System.out.println(str);

            return;

        }

        for(int i=low;i<=high;i++){

            swap(str,low,i);

            permu(str,low+1,high);

            swap(str,low,i);

        }

    }

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

        Scanner sc=new Scanner(System.in);

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

        String str=sc.nextLine();

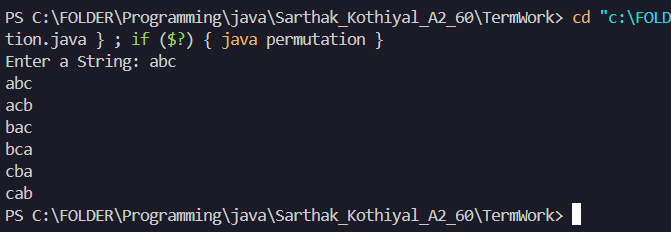
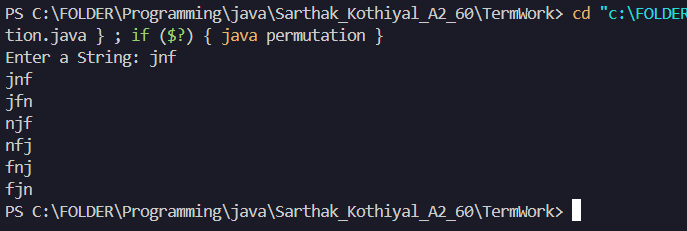
        StringBuffer s=new StringBuffer(str);

        permu(s,0,str.length()-1);

    }

}

**Output:**

**  
  
  
  
**

**Practical No. 7:**

Find out if there are any occurrences of the word "city" in a sentence

**Source Code:**

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class citycheck {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string:");

String str = sc.nextLine();

Pattern pattern = Pattern.compile("\\bcity\\b", Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher(str);

int count = 0;

while (matcher.find()) {

count++;

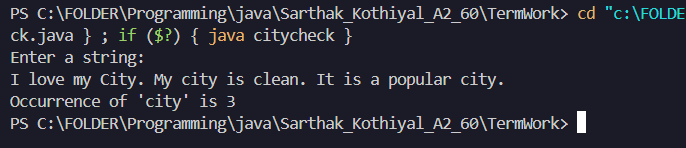
}

System.out.println("Occurrence of 'city' is " + count);

}

}

**Output:**

****

**Practical No. 8:**

Check if Email Address is Valid or not in Java.

**Source Code:**

import java.util.Scanner;

public class validemail {

public static boolean check(String email) {

if (email == null){

return false;

}

int atPos = email.indexOf('@');

if (atPos <= 0){

return false;

}

int lastAtPos = email.lastIndexOf('@');

if (atPos != lastAtPos){

return false;

}

int dotPos = email.indexOf('.', atPos);

if (dotPos == -1 || dotPos == atPos + 1){

return false;

}

if (email.endsWith(".")){

return false;

}

if (email.contains(" ")){

return false;

}

return true;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter email address: ");

String email = sc.nextLine();

if (check(email)) {

System.out.println("Email is valid.");

}

else {

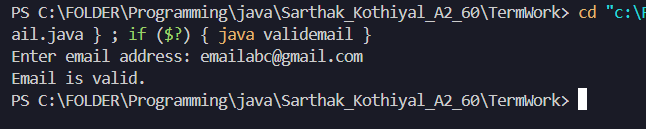
System.out.println("Email is NOT valid.");

}

}

}

**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.\*;

public class minplatforms {

    private static int toMinutes(String *time*) {

        String[] parts = time.split(":");

        return Integer.parseInt(parts[0]) \* 60 + Integer.parseInt(parts[1]);

    }

    public static int findMinimumPlatforms(String[] *arr*, String[] *dep*) {

        int n = arr.length;

        int[] arrival = new int[n];

        int[] departure = new int[n];

        for (int i = 0; i < n; i++) {

            arrival[i] = toMinutes(arr[i]);

            departure[i] = toMinutes(dep[i]);

        }

        Arrays.sort(arrival);

        Arrays.sort(departure);

        int platformsNeeded = 0, maxPlatforms = 0;

        int i = 0, j = 0;

        while (i < n && j < n) {

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

                platformsNeeded++;

                i++;

            }

            else {

                platformsNeeded--;

                j++;

            }

            maxPlatforms = Math.max(maxPlatforms, platformsNeeded);

        }

        return maxPlatforms;

    }

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

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of trains: ");

        int n = sc.nextInt();

        sc.nextLine();

        String[] arrivals = new String[n];

        String[] departures = new String[n];

        System.out.println("Enter arrival time:");

        for (int i = 0; i < n; i++) {

            arrivals[i] = sc.nextLine();

        }

        System.out.println("Enter departure time:");

        for (int i = 0; i < n; i++) {

            departures[i] = sc.nextLine();

        }

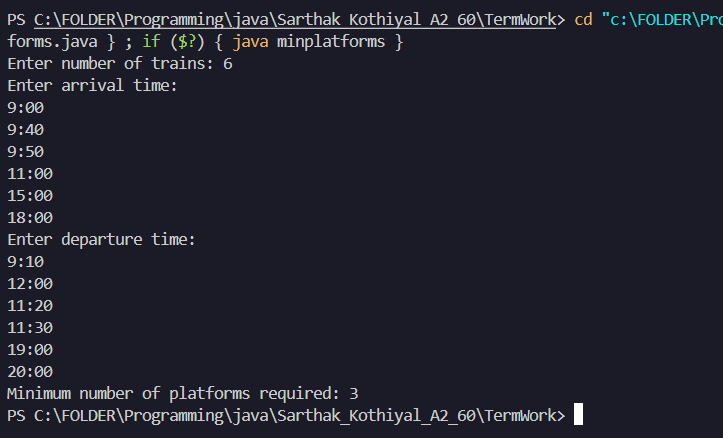
        int result = findMinimumPlatforms(arrivals, departures);

        System.out.println("Minimum number of platforms required: " + result);

    }

}

**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.Scanner;

public class waveform {

    public static void convertToWave(int[] *arr*) {

        int n = arr.length;

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

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

                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 number of elements: ");

        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter elements:");

        for (int i = 0; i < n; i++) {

            arr[i] = sc.nextInt();

        }

        convertToWave(arr);

        System.out.println("Wave form of the array:");

        for (int num : arr) {

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

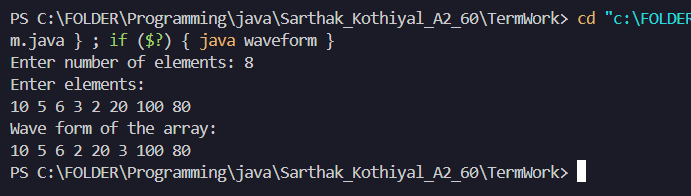
        }

        sc.close();

    }

}

**Output:**

****