DSA PRACTICE PROBLEM DAY 6

PROBLEM 1 :

BUBBLE SORT:

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

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

int n = scan.nextInt();

int[] arr = new int[n];

System.out.println("Enter the Elements in the Array :");

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

arr[i] = scan.nextInt();

}

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

boolean swap = false;

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

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

int temp=arr[j];

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

arr[j+1] = temp;

swap = true;

}

}

if(!swap){

break;

}

}

for(int i:arr){

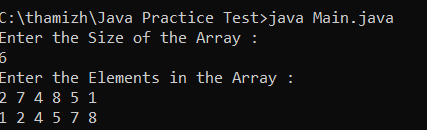
System.out.print(i+" ");

}

}

}

OUTPUT;



PROBLEM 2:

QUICK SORT:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

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

int n = scan.nextInt();

int[] arr = new int[n];

System.out.println("Enter the Elements in the Array :");

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

arr[i] = scan.nextInt();

}

int low = 0;

int high = n-1;

quickSort(arr,low,high);

for(int i:arr){

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

}

}

static void quickSort(int arr[], int low, int high) {

if (low < high) {

int partitionIndex = partition(arr, low, high);

quickSort(arr, low, partitionIndex - 1);

quickSort(arr, partitionIndex + 1, high);

}

}

static int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = low - 1;

for (int j = low; j < high; j++) {

if (arr[j] <= pivot) {

i++;

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

int temp = arr[i + 1];

arr[i + 1] = arr[high];

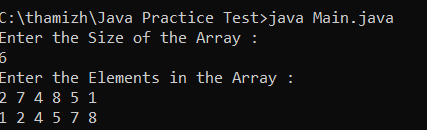
arr[high] = temp;

return i + 1;

}

}

OUTPUT:



PROBLEM 3:

Given a string s consisting of lowercase Latin Letters. Return the first non-repeating character in s. If there is no non-repeating character, return '$'.

Note: When you return '$' driver code will output -1.

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

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

String s = scan.next();

char result = '$';

boolean found = false;

for(char ch : s.toCharArray()){

if(s.indexOf(ch)==s.lastIndexOf(ch)){

result = ch;

found = true;

break;

}

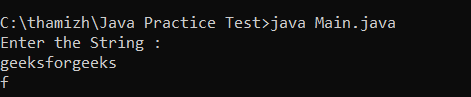
}

System.out.println(result);

}

}

OUTPUT :



PROBLEM 4:

EDIT DISTANCE :

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the String1 :");

String word1 = scan.next();

System.out.println("Enter the String2 :");

String word2 = scan.next();

int m = word1.length();

int n = word2.length();

int[] prev = new int[m + 1];

int[] curr = new int[m + 1];

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

prev[i] = i;

}

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

curr[0] = j;

for (int i = 1; i <= m; i++) {

if (word1.charAt(i - 1) == word2.charAt(j - 1)) {

curr[i] = prev[i - 1];

} else {

int replace = prev[i - 1];

int delete = prev[i];

int insert = curr[i - 1];

curr[i] = Math.min(replace, Math.min(delete, insert)) + 1;

}

}

int[] temp = prev;

prev = curr;

curr = temp;

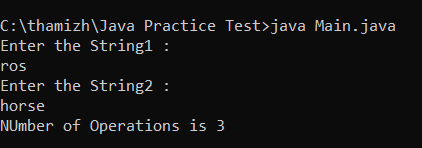
}

System.out.println("NUmber of Operations is " + prev[m]);

}

}

OUTPUT :



PROBLEM 5:

Given an array arr[] of positive integers and an integer k, Your task is to return k largest elements in decreasing order.

CODE ;

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the size of the array :");

int n = scan.nextInt();

int[] arr = new int[n];

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

arr[i] = scan.nextInt();

}

System.out.println("Enter the valure of K ");

int k = scan.nextInt();

List<Integer> result = new ArrayList<>();

Arrays.sort(arr);

int i = 0;

int j = arr.length-1;

while(i<k){

result.add(arr[j]);

i++;

j--;

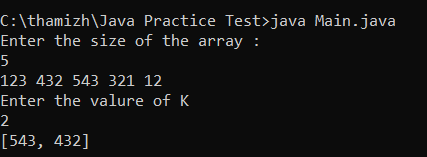
}

System.out.println(result);

}

}

OUTPUT :



PROBLEM 6:

Given an array of integers arr[] representing non-negative integers, arrange them so that after concatenating all of them in order, it results in the largest possible number. Since the result may be very large, return it as a string.

Note: There are no leading zeros in each array element.

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the size of the array :");

int n = scan.nextInt();

int[] nums = new int[n];

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

nums[i] = scan.nextInt();

}

String[] s = new String[nums.length];

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

s[i] = String.valueOf(nums[i]);

}

Arrays.sort(s,(a,b)->(b+a).compareTo(a+b));

System.out.println(s[0].equals("0")?"0":String.join("",s));

}

}

OUTPUT:

