1------------------

**//input String str ="aabbbcca";**

**//output a2b3c2a1**

public class TestNew {

public static void main(String[] args) {

String str = "aabbbcca";

int count = 0;

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

count = 1;

for (int j = i + 1; j < str.length(); j++) {

if (str.charAt(i) == str.charAt(j)) {

count++;

} else {

break;

}

}

System.out.print(str.charAt(i) +""+ count);

}

}

}

**2--------remove duplicate element from array and print it.**

// input { 1, 2, 3, 3, 4, 4, 5 }

// output 1 2 3 4 5

public class TestNew {

public static void main(String[] args) {

int[] arr = { 1, 2, 3, 3, 4, 4, 5 };

int k = 0;

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

if (arr[i] != arr[i + 1]) {

arr[k++] = arr[i];

}

}

arr[k++] = arr[arr.length - 1];

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

System.out.print(arr[i] + " ");

}

}

}

**3---- seperate 0 and 1 in array**

public class TestNew {

public static void main(String[] args) {

int[] arr = { 1, 0, 0, 0, 1, 1, 0, 1 };

int count = 0;

for (int val : arr) {

if (val == 0) {

count = count + 1;

}

}

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

if (i < count) {

arr[i] = 0;

} else {

arr[i] = 1;

}

}

for (int val : arr) {

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

}

}

}

**4-------------------seperate positive and negative number**

public class TestNew {

public static void main(String[] args) {

int[] arr = { 1, 2, 5, -1, -6, 7 };

int[] temp = new int[arr.length];

int j = 0;

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

if (arr[i] > 0) {

temp[j++] = arr[i];

}

}

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

if (arr[i] < 0) {

temp[j++] = arr[i];

}

}

for (int val : temp) {

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

}

}

}

**5----------------------delete element from array**

// input { 1, 2, 3, 4, 5, 6 }

// output {1,2,3,5,6}

public class TestNew {

public static void main(String[] args) {

int[] arr = { 1, 2, 3, 4, 5, 6 };

int[] newArr = new int[arr.length-1];

int j=0;

int k=3;

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

newArr[j++]=arr[i];

}

for(int i=k+1;i<arr.length;i++) {

newArr[j++]=arr[i];

}

for(int val : newArr) {

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

}

}

}

**6--------------------- insert an element into array**

// input { 1, 2, 3, 4, 5, 6 }

// output {1,2,3,5,6}

public class TestNew {

public static void main(String[] args) {

int[] arr = { 1, 2, 3, 4, 5, 6 };

int[] newArr = new int[arr.length + 1];

int j = 0;

int element = 11;

int index = 4;

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

newArr[j++] = arr[i];

}

newArr[j++] = element;

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

newArr[j++] = arr[i];

}

for (int val : newArr) {

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

}

}

}

**7 ) merge two sorted array**

public class TestNew {

public static void main(String[] args) {

int[] arr1 = { 1, 2, 3, 7 };

int[] arr2 = { 4, 5, 9, 10 };

int[] newArr = new int[arr1.length + arr2.length];

int i = 0, j = 0, k = 0;

while (i < arr1.length && j < arr2.length) {

if (arr1[i] < arr2[j]) {

newArr[k++] = arr1[i++];

} else {

newArr[k++] = arr2[j++];

}

}

while (i < arr1.length) {

newArr[k++] = arr1[i++];

}

while (j < arr2.length) {

newArr[k++] = arr2[j++];

}

for (int val : newArr) {

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

}

}

}

public static void main(String[] args) {

int arr1[] = { 1, 3, 5, 7 };

int arr2[] = { 2, 4, 6, 8 };

int arr3[] = mergeArrays(arr1, arr2);

System.out.println("Array after merging - " + Arrays.toString(arr3));

}

public static int[] mergeArrays(int[] arr1, int[] arr2) {

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

for (int val : arr1) {

if (val != 0) {

list.add(val);

}

}

for (int val : arr2) {

if (val != 0) {

list.add(val);

}

}

List<Integer> sortedList = list.stream().sorted().collect(Collectors.toList());

System.out.println(sortedList);

int size = sortedList.size();

int[] newArr = new int[size];

int k=0;

for( int val : sortedList) {

newArr[k++]=val;

}

return newArr;

}

// output

**8---- find maximum product of two integer in an array**

**// input { 1, 2, 3, 4, 5, 6 }**

**// output {1,2,3,5,6}**

public class TestNew {

public static void main(String[] args) {

int[] arr1 = { 1, 2, 3, 7 };

int maxProduct = Integer.MIN\_VALUE;

for (int i = 0; i < arr1.length - 1; i++) {

for (int j = i + 1; j < arr1.length; j++) {

int prod = arr1[i] \* arr1[j];

if (prod > maxProduct) {

maxProduct = prod;

}

}

}

System.out.println(maxProduct);

}

}

**9-------------- find intersection between two array**

**// input**

**//int[] arr1 = { 1, 2, 3, 7 };**

**//int[] arr2 = { 2, 7, 9 };**

// output 2,7

public class TestNew {

public static void main(String[] args) {

int[] arr1 = { 1, 2, 3, 7 };

int[] arr2 = { 2, 7, 9 };

int l = arr1.length;

int j = arr2.length;

int k = 0;

if (l > j) {

k = j;

} else {

k = l;

}

int index = 0;

int[] intersection = new int[k];

for (int i = 0; i < arr1.length; i++) {

int current = arr1[i];

// Check if current element exists in arr2

if (contain(arr2, current)) {

// Add to intersection array if it's not already added

if (!contain(intersection, current)) {

intersection[index++] = current;

}

}

}

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

System.out.print(intersection[i]+" ");

}

}

private static boolean contain(int[] arr2, int val) {

for (int i = 0; i < arr2.length; i++) {

if (arr2[i] == val) {

return true;

}

}

return false;

}

}

10-

**Write a Java program to print a sequence of numbers up to N using 3 threads. For example, if we want to print a sequence of numbers up to 10 then it’ll look like this: THREAD-1 : 1 THREAD-2 : 2 THREAD-3 : 3 THREAD-1 : 4 THREAD-2 : 5 THREAD-3 : 6 THREAD-1 : 7 THREAD-2 : 8 THREAD-3 : 9 THREAD-1 : 10**

package com;

public class ThreadSequence {

private static final int N = 10;

private static int currentNumber = 1;

private static final Object lock = new Object();

private static int threadTurn = 1;

public static void main(String[] args) {

// Create three threads with instances of PrintTask

Thread t1 = new Thread(new PrintTask(1), "THREAD-1");

Thread t2 = new Thread(new PrintTask(2), "THREAD-2");

Thread t3 = new Thread(new PrintTask(3), "THREAD-3");

// Start all three threads

t1.start();

t2.start();

t3.start();

}

// PrintTask class implementing Runnable for each thread

static class PrintTask implements Runnable {

private final int threadId;

public PrintTask(int threadId) {

this.threadId = threadId;

}

@Override

public void run() {

while (currentNumber <= N) {

synchronized (lock) {

// Check if it's this thread's turn to print

while (threadTurn != threadId) {

try {

lock.wait(); // Wait if it's not this thread's turn

} catch (InterruptedException e) {

e.printStackTrace();

}

}

// Print the current number and thread name

System.out.println(Thread.currentThread().getName() + " : " + currentNumber);

currentNumber++;

// Update threadTurn to the next thread in round-robin fashion

threadTurn = (threadTurn % 3) + 1;

// Notify all threads waiting on the lock

lock.notifyAll();

}

}

}

}

}

**11---even and odd using thread**

package com;

public class EvenOddDemo {

private static int n = 10;

private static Object lock = new Object();

private static int count = 1;

public static void main(String[] args) {

Thread even = new Thread(new EvenCheck());

Thread odd = new Thread(new OddCheck());

even.start();

odd.start();

}

static class EvenCheck implements Runnable {

@Override

public void run() {

while (count <= n) {

synchronized (lock) {

while ( count % 2 != 0) {

try {

lock.wait(); // wait until it is an even number's turn

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println("EVEN THREAD: " + count);

count++;

lock.notifyAll(); // notify other waiting threads

}

}

}

}

static class OddCheck implements Runnable {

@Override

public void run() {

while (count <= n) {

synchronized (lock) {

while ( count % 2 == 0) {

try {

lock.wait(); // wait until it is an even number's turn

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println("Odd THREAD: " + count);

count++;

lock.notifyAll(); // notify other waiting threads

}

}

}

}}

**12-----------:**

**Remove some duplicates in-place such that each unique element appears at most twice Example - Input: nums = [1,1,1,2,2,3] Output: 5, nums = [1,1,2,2,3,null]**

Using with one array

public class Demo

{

public static void main(String[] args) {

int arr[] = { 1, 2, 2, 2, 4, 4, 4, 5, 5 };

int n = arr.length;

// removeDuplicates() returns new size of array

n = removeDuplicates(arr, n);

// Print updated array

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

System.out.print(arr[i] + " ");

}

static int removeDuplicates(int arr[], int n){

if(arr.length<2){

return 0;

}

int k=0;

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

if(i<arr.length-2 &&arr[i]==arr[i+1] &&arr[i+1]==arr[i+2]) {

continue;

}else {

arr[k++]=arr[i];

}

}

return k;

}

}

**Question 13**

**Q: You are given an array, Print the output as below:**

**[0, 1, 2]**

**[3, 4, 5]**

**[6, 7, 8]**

**[9]**

Solution

public class Demo {

public static void main(String[] args) {

int[] array = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int elementsPerLine = 3;

int lines = array.length / elementsPerLine;

int remainder = array.length % elementsPerLine;

int index = 0;

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

System.out.print("[");

for (int j = 0; j < elementsPerLine; j++) {

System.out.print(array[index++]);

if (j < elementsPerLine - 1) {

System.out.print(", ");

}

}

System.out.println("]");

}

if (remainder > 0) {

System.out.print("[");

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

System.out.print(array[index++]);

if (i < remainder - 1) {

System.out.print(", ");

}

}

System.out.println("]");

}

}

}

**Question 14 Write a program to print the characters which are matching in both the strings**

**package dom;**

import java.util.HashSet;

import java.util.Set;

public class Test

{

public static void main(String[] args) {

String str1 = "programming";

String str2 = "language";

// Step 1: Count frequency of characters in str1 using HashMap

Set<Character> set = new HashSet<>();

for(char ch :str1.toCharArray()) {

set.add(ch);

}

for(char ch : str2.toCharArray()) {

if(set.contains(ch)) {

System.out.println(ch);

set.remove(ch);

}

}

}

}

public class TestNew {

public static void main(String[] args) {

String str1 = "programming";

String str2 = "language";

boolean[] visited = new boolean[256];

for (char ch : str1.toCharArray()) {

visited[ch] = true;

}

for (char ch : str2.toCharArray()) {

if (visited[ch]) {

System.out.println(ch);

visited[ch] = false;

}

}

}

}

**15------------binary search**

public class BinarySearch {

private static void binarySearch(int[] a, int search) {

int low = 0;

int high = a.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (a[mid] > search) {

high = mid - 1;

} else if (a[mid] < search) {

low = mid + 1;

} else {

System.out.println("found element at index " + mid);

break;

}

System.out.println("value not found");

}

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int[] a = new int[n];

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

a[i] = s1.nextInt();

}

int search = s1.nextInt();

binarySearch(a, search);

}

}

**16 : pattern use star and space concept**

**/\* output**

**\* \* \* \* \***

**\* \* \* \***

**\* \* \***

\* \*

\*

\*/

public class Pattern4 {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int star = n;

int space = 0;

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

for (int k = 1; k <= space; k++) {

System.out.print("\t");

}

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

System.out.print("\*\t");

}

space++;

star--;

System.out.println();

}

}

}

**17- pattern**

**/\***

**\***

**\* \* \***

**\* \* \* \* \***

**\* \* \***

**\***

**\*/**

public class Pattern5 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int sp = n / 2;

int st = 1;

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

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

System.out.print("\t");

}

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

System.out.print("\*\t");

}

if (i <= n / 2) {

sp--;

st = st + 2;

} else {

sp++;

st = st - 2;

}

System.out.println();

}

}

}

**18 pattern**

**\* output**

**\* \* \* \* \* \***

**\* \* \* \***

**\* \***

**\* \* \* \***

**\* \* \* \* \* \***

**\*/**

public class Pattern6 {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int sp = 1;

int st = n / 2 + 1;

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

for(int j=1;j<=st;j++)

{

System.out.print("\*\t");

}

for(int j=1;j<=sp;j++)

{

System.out.print("\t");

}

for(int j=1;j<=st;j++)

{

System.out.print("\*\t");

}

if (i <= n / 2) {

sp += 2;

st--;

} else {

sp -= 2;

st++;

}

System.out.println();

}

}

}

19 pattern

import java.util.Scanner;

/\*output

\*

\*

\*

\*

\*

\*/

public class Pattern8 {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int sp = n - 1;

int st = 1;

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

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

System.out.print("\t");

}

for (int k = 1; k <= st; k++) {

System.out.print("\*\t");

}

sp--;

System.out.println();

}

}

**20 pattern**

**public class Pattern9 {**

**/\* output**

**\* \***

**\* \***

**\***

**\* \***

**\* \***

**\*/**

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

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

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

if (i == j || i + j == n + 1) {

System.out.print("\*\t");

} else {

System.out.print("\t");

}

}

System.out.println();

}

}

}

**21 pattern**

**import java.util.Scanner;**

**/\***

**1**

**2 3**

**4 5 6**

**7 8 9 10**

**11 12 13 14 15**

**\*/**

public class Pattern11 {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int val = 1;

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

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

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

val++;

}

System.out.println();

}

}

}

**22- pattern**

**/\* output in fibonacci series**

**0**

**1 1**

**2 3 5**

**8 13 21 34**

**55 89 144 233 377**

\*/

public class Pattern12 {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int a = 0;

int b = 1;

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

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

System.out.print(a + "\t");

int c = a + b;

a = b;

b = c;

}

System.out.println();

}

}

}

**23--- reverse number**

public class ReverseNumber {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int number = s1.nextInt();

while (number > 0) {

int remainder = number % 10;

System.out.print(remainder);

number = number / 10;

}

}

}

**24---didgit frquency**

**/\***

**input 1234 1**

**output 1**

**\*/**

public class DigitFrequency {

private static int getDigitFrequency(int n, int d) {

int count = 0;

while (n != 0) {

int rem = n % 10;

if (rem == d) {

count++;

}

n = n / 10;

}

return count;

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int d = s1.nextInt();

int f =getDigitFrequency(n, d);

System.out.println(f);

}

}

**25 : rotate and array**

**/\***

**input n=4**

**1**

**2**

**3**

**4**

**k=2**

**output 3 4 1 2**

**\*/**

public class RotateAnArray {

private static void reverseAnArray(int[] a,int i, int j) {

while (i < j) {

int temp = a[i];

a[i] = a[j];

a[j] = temp;

i++;

j--;

}

}

private static void rotateArray(int[] a, int k) {

k = k % a.length;

if (k < 0) {

k = k + a.length;

}

reverseAnArray(a, 0, a.length - k - 1);

reverseAnArray(a, a.length - k, a.length - 1);

reverseAnArray(a, 0, a.length - 1);

}

private static void display(int[] a) {

for (int a1 : a) {

System.out.print(a1 + " ");

}

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int[] a = new int[n];

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

a[i] = s1.nextInt();

}

int k = s1.nextInt();

rotateArray(a, k);

display(a);

}

}

**26--- find ceil floor**

**import java.util.Scanner;**

**/\***

**input 5**

**11**

**22**

**33**

**44**

**55**

**element to search 34**

**output ceil is 44 and floor is 33**

**\*/**

public class CeilFloor {

private static void ceilFloor(int[] a, int search) {

int low = 0;

int high = a.length - 1;

int ceil = 0;

int floor = 0;

while (low <= high) {

int mid = (low + high) / 2;

if (a[mid] > search) {

high = mid - 1;

ceil = a[mid];

} else if (a[mid] < search) {

low = mid + 1;

floor = a[mid];

} else {

ceil = a[mid];

floor = a[mid];

break;

}

}

System.out.println("ceil is " + ceil + " and floor is " + floor);

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int[] a = new int[n];

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

a[i] = s1.nextInt();

}

int search = s1.nextInt();

ceilFloor(a, search);

}

}

**27 ---- span of an array ( max-min)**

**/\***

**input**

**5**

**12**

**1**

**2**

**4**

**8**

**output 11**

**\*/**

public class Span\_Of\_Array {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int[] arr = new int[n];

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

arr[i] = s1.nextInt();

}

int max = arr[0];

int min = arr[0];

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

if (arr[j] > max) {

max = arr[j];

}

if (arr[j] < min) {

min = arr[j];

}

}

int span = max - min;

System.out.println(span);

}

}

**28 : remove element at index :**

public class removeElementAtIndex {

public static void main(String[] args) {

int[] arr = { 1, 2, 3, 4, 5, 6, 7 };

int index = 4;

int[] removeElement = removeElement(arr,index);

for(int val : removeElement) {

System.out.println(val);

}

}

private static int[] removeElement(int[] arr, int index) {

int[] newArray = new int[arr.length-1];

int j=0;

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

if(i==index) {

continue;

}

newArray[j++]=arr[i];

}

return newArray;

}

}

**29 :--- sort map using values as grade**

first store it into list as map of entry then sort it using collections and store it into linkedhashmap

package com;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Map.Entry;

public class TestNew {

public static void main(String[] args) {

HashMap<String, String> hm = new HashMap<>();

hm.put("test", "A");

hm.put("hello", "B+");

hm.put("rahul", "C");

hm.put("meena", "d");

ArrayList<Map.Entry<String, String>> list = new ArrayList<>(hm.entrySet());

Collections.sort(list, new Comparator<Map.Entry<String, String>>() {

@Override

public int compare(Entry<String, String> o1, Entry<String, String> o2) {

// TODO Auto-generated method stub

return o2.getValue().compareTo(o1.getValue());

}

});

Map<String, String> linkedHashMap = new LinkedHashMap<>();

for (Map.Entry<String, String> val : list) {

linkedHashMap.put(val.getKey(), val.getValue());

}

for (Map.Entry<String, String> val : linkedHashMap.entrySet()) {

System.out.println(val.getKey() + " " + val.getValue());

}

}

}

**30---------------------stack**

package com;

public class TestNew {

int[] data;

int tos;

public TestNew(int n) {

data = new int[n];

tos = -1;

}

public int size() {

return tos + 1;

}

public void push(int val) {

if (tos == data.length - 1) {

System.out.println("stack is overflow");

} else {

tos++;

data[tos] = val;

}

}

public int pop() {

if (tos == -1) {

System.out.println("stack is underflow");

return -1;

} else {

int val = data[tos];

tos--;

return val;

}

}

int top() {

if (tos == -1) {

System.out.println("stack underflow");

return -1;

} else {

int val = data[tos];

return val;

}

}

public void display() {

for (int i = tos; i >= 0; i--) {

System.out.println(data[i] + " ");

}

}

public static void main(String[] args) {

TestNew t = new TestNew(5);

System.out.println("push");

t.push(1);

t.push(2);

t.push(3);

t.display();

System.out.println("pop");

t.pop();

t.display();

System.out.println("top");

t.top();

t.display();

}

}

// output

push

3

2

1

pop

2

1

top

2

1

**31 print decreasing**

**import java.util.Scanner;**

**/\***

**\* input : 5**

**\* output**

**5**

**4**

**3**

**2**

**1**

**\*/**

public class PrintDecreasing {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

printDecreasing(n);

}

private static void printDecreasing(int n) {

if (n == 0) {

return;

}

System.out.println(n);

printDecreasing(n - 1);

}

}

**32) print increasing**

**import java.util.Scanner;**

**/\***

**input 5**

**output**

**1**

**2**

**3**

**4**

**5**

**\*/**

public class PrintIncreasing {

private static void printIncreasing(int n) {

if(n==0)

{

return;

}

printIncreasing(n-1);

System.out.println(n);

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n=s1.nextInt();

printIncreasing(n);

}

}

**33 :) print decreasing and increasing**

**import java.util.Scanner;**

**/\***

**input 2**

**output**

**2**

**1**

**1**

**2**

**\*/**

public class PrintDecreasingIncreasing {

private static void printDI(int n) {

if(n==0)

{

return;

}

System.out.println(n);

printDI(n-1);

System.out.println(n);

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n=s1.nextInt();

printDI(n);

}

}

**34 :--- reverse string using recursion :-**

public class TestNew {

public static void main(String[] args) {

String name = "neha";

String reverse = reverse(name);

System.out.println(reverse);

}

private static String reverse(String name) {

if (name.length() == 0) {

return "";

}

return reverse(name.substring(1)) + name.charAt(0);

}

}

**35 :--- string reverse**

public class TestNew {

public static void main(String[] args) {

String name = "saurabh";

char[] ch = name.toCharArray();

int i = 0;

int j = name.length() - 1;

while (i < j) {

char temp = ch[i];

ch[i] = ch[j];

ch[j] = ch[i];

i++;

j--;

}

String reverseString = String.valueOf(ch);

System.out.println(reverseString);

}

}

===============================or=======================

public class TestNew {

public static void main(String[] args) {

String name = "saurabh";

String reverse="";

for(int i = name.length() - 1; i >= 0; i--) {

reverse = reverse + name.charAt(i);

}

System.out.println(reverse);

}

}

====================================

**36 :--- print prime number**

public class TestNew {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

boolean flag = true;

for (int div = 2; div \* div <= n; div++) {

if (n % div == 0) {

flag = false;

break;

}

}

if (flag) {

System.out.println(n + " is prime number");

} else {

System.out.println(n + " is not prime number");

}

}

}

37--- print all prime number

ublic class PrimeNumbersInRange {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the lower bound (low): ");

int low = scanner.nextInt();

System.out.print("Enter the upper bound (high): ");

int high = scanner.nextInt();

if (low < 0) {

low = 2; // Start from 2 if low is negative (considering 0 and 1 are not prime numbers)

}

System.out.println("Prime numbers between " + low + " and " + high + ":");

for (int i = Math.max(low, 2); i <= high; i++) {

boolean isPrime = true;

// Check for prime

for (int div = 2; div \* div <= i; div++) {

if (i % div == 0) {

isPrime = false;

break;

}

}

if (isPrime) {

System.out.println(i);

}

}

}

}

**38 ----- anagram program**

public class AnagramChecker {

public static boolean areAnagrams(String str1, String str2) {

// Remove spaces and convert to lowercase

String s1 = str1.replaceAll("\\s", "").toLowerCase();

String s2 = str2.replaceAll("\\s", "").toLowerCase();

// Check if lengths are equal

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

return false;

}

// Convert strings to char arrays, sort them, and compare

return Arrays.equals(s1.chars().sorted().toArray(), s2.chars().sorted().toArray());

}

public static void main(String[] args) {

String str1 = "Listen";

String str2 = "Silent";

if (areAnagrams(str1, str2)) {

System.out.println(str1 + " and " + str2 + " are anagrams.");

} else {

System.out.println(str1 + " and " + str2 + " are not anagrams.");

}

}

}

**----------- decreasing using recursion**

**39) display array decreasing order**

import java.util.Scanner;

/\*

input no of element 3- 1,2,3

output

1

2

3

\*/

public class DisplayArrayReverse {

private static void printArray(int[] arr, int i) {

if (arr.length == i) {

return;

}

printArray(arr, i + 1);

System.out.println(arr[i]);

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int[] arr = new int[n];

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

arr[i] = s1.nextInt();

}

printArray(arr, 0);

}

}

**40---- display array using recursion in increasing order**

display array increasing order

import java.util.Scanner;

/\*

Input : no of element 3 : 1,2 ,3

output

1

2

3

\*/

public class DisplayArray {

private static void printArray(int[] arr, int i) {

if (i == arr.length) {

return;

}

System.out.println(arr[i]);

printArray(arr, i + 1);

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int[] arr = new int[n];

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

arr[i] = s1.nextInt();

}

printArray(arr, 0);

}

}

**41 -------------- factorial using recursion**

public class TestNew {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

int fact = fact(n);

System.out.println(fact);

}

private static int fact(int n) {

if (n == 1) {

return 1;

}

return n \* fact(n - 1);

}

}

42) print 2d array

/\*

input

row :- 3

column :-3

element

11

12

13

22

23

24

33

34

35

output

11 12 13

22 23 24

33 34 35

\*/

public class TwoDArray {

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int r = s1.nextInt();

int c = s1.nextInt();

int[][] arr = new int[r][c];

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

for (int j = 0; j < c; j++) {

arr[i][j] = s1.nextInt();

}

}

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

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

System.out.print(arr[i][j] + "\t");

}

System.out.println();

}

}

}

**43-**

**print all pailandrome**

**import java.util.Scanner;**

**/\***

**\* input : abac**

**\* output**

**\* a**

**aba**

**b**

**a**

c

\*/

public class printAllPalindromeSubstring {

private static void solution(String str) {

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

for (int j = i + 1; j <= str.length(); j++) {

String ss = str.substring(i, j);

if (isPalindrome(ss) == true) {

System.out.println(ss);

}

}

}

}

private static boolean isPalindrome(String ss) {

int i = 0;

int j = ss.length() - 1;

while (i <= j) {

char ch1 = ss.charAt(i);

char ch2 = ss.charAt(j);

if (ch1 != ch2) {

return false;

} else {

i++;

j--;

}

}

return true;

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

String str = s1.next();

solution(str);

}

}

**44- string compressed :-**

public class TestNew {

public static void main(String[] args) {

String name = "aabbcccdee";

String str = name.charAt(0) + "";

for (int i = 1; i < name.length(); i++) {

char cur = name.charAt(i);

char prev = name.charAt(i - 1);

if (cur != prev) {

str = str + cur;

}

}

System.out.println(str);

}

}

45 : remove prime array list

/\*

input 5

11

12

13

14

15

output

[12, 14, 15]

\*/

public class RemovePrimeArrayList {

private static void removePrime(ArrayList<Integer> al) {

for (int i = al.size() - 1; i >= 0; i--) {

int val = al.get(i);

if (isPrime(val) == true) {

al.remove(i);

}

}

}

private static boolean isPrime(int val) {

for (int i = 2; i \* i <= val; i++) {

if (val % i == 0) {

return false;

}

}

return true;

}

public static void main(String[] args) {

Scanner s1 = new Scanner(System.in);

int n = s1.nextInt();

ArrayList<Integer> al = new ArrayList<Integer>();

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

al.add(s1.nextInt());

}

removePrime(al);

System.out.println(al);

}

}

**46---- expression balanced using stack**

package com;

import java.util.Stack;

public class TestNew {

public static void main(String[] args) {

String str = "[{(}]";

boolean checkBalanced = checkBalanced(str);

if (checkBalanced) {

System.out.println("expression is balanced");

} else {

System.out.println("expression is not balanced");

}

}

private static boolean checkBalanced(String str) {

Stack<Character> st = new Stack();

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

char ch = str.charAt(i);

if (ch == '(' || ch == '{' || ch == '[') {

st.push(ch);

} else if (ch == ')' || ch == '}' || ch == ']') {

if (st.isEmpty()) {

return false;

}

char top = st.pop();

if ((ch == ')' && top != '(') || (ch == '{' && top != '}') || (ch == '[' && top != ']')) {

return false;

}

}

}

return st.isEmpty();

}

}

**47----------- immutable class**

package com;

import java.util.Date;

public final class ImmutableClass {

private final String name;

private final int age;

private final Date birthDate; // Date is mutable

public ImmutableClass(String name, int age, Date birthDate) {

this.name = name;

this.age = age;

// Creating a new Date object to ensure the immutability

this.birthDate = new Date(birthDate.getTime());

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

public Date getBirthDate() {

// Returning a new Date object to maintain immutability

return new Date(birthDate.getTime());

}

@Override

public String toString() {

return "ImmutableClass [name=" + name + ", age=" + age + ", birthDate=" + birthDate + "]";

}

public static void main(String[] args) {

Date date = new Date();

ImmutableClass obj = new ImmutableClass("John Doe", 30, date);

System.out.println("Before changing date: " + obj);

// Trying to modify the birthDate outside the class

date.setTime(date.getTime() + 1000000000L);

System.out.println("After changing date: " + obj);

// Trying to modify the birthDate using the getter method

obj.getBirthDate().setTime(date.getTime() + 1000000000L);

System.out.println("After modifying date from getter: " + obj);

}

}

48:--

package com;

public class Subject {

private String name;

private double marks;

public Subject(String name, double marks) {

this.name = name;

this.marks = marks;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getMarks() {

return marks;

}

public void setMarks(double marks) {

this.marks = marks;

}

@Override

public String toString() {

return "Subject [name=" + name + ", marks=" + marks + "]";

}

}

class Person {

private String name;

private List<Subject> list;

public Person(String name, List<Subject> list) {

this.name = name;

this.list = list;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public List<Subject> getList() {

return list;

}

public void setList(List<Subject> list) {

this.list = list;

}

@Override

public String toString() {

return "Person [name=" + name + ", list=" + list + "]";

}

}

public class TestNew {

public static void main(String[] args) {

// Math 80 english 90 science 88

List<Subject> list = new ArrayList<>();

Subject s1 = new Subject("Math", 80.0);

Subject s2 = new Subject("English", 90.0);

Subject s3 = new Subject("Science", 88.0);

list.add(s1);

list.add(s2);

list.add(s3);

List<Subject> list1 = new ArrayList<>();

Subject s4 = new Subject("Math", 90.0);

Subject s5 = new Subject("English", 50.0);

Subject s6 = new Subject("Science", 88.0);

list1.add(s4);

list1.add(s5);

list1.add(s6);

List<Subject> list2 = new ArrayList<>();

Subject s7 = new Subject("Math", 50.0);

Subject s8 = new Subject("English", 70.0);

Subject s9 = new Subject("Science", 98.0);

list2.add(s7);

list2.add(s8);

list2.add(s9);

Person p1 = new Person("A", list);

Person p2 = new Person("B", list1);

Person p3 = new Person("C", list2);

// /

List<Person> persons = new ArrayList<>();

persons.add(p1);

persons.add(p2);

persons.add(p3);

for (String subjectName : new String[] { "Math", "English", "Science" }) {

String topPerson = "";

double highestMarks = -1;

for (Person person : persons) {

for (Subject subject : person.getList()) {

if (subject.getName().equals(subjectName) && subject.getMarks() > highestMarks) {

highestMarks = subject.getMarks();

topPerson = person.getName();

}

}

}

System.out.println("Highest marks in " + subjectName + " are by " + topPerson + " with " + highestMarks);

}

}

}

**49 :----next greater element**

public class TestProg {

public static void main(String[] args) {

int[] arr = { 1, 8, 7, 9, 3, 5, 6, 12, 4 };

//output 8 9 9 12 5 6 12 -1 -1

nextGreaterElement(arr);

}

private static void nextGreaterElement(int[] arr) {

Stack<Integer> st = new Stack<>();

int[] nge = new int[arr.length];

st.push(arr[arr.length - 1]);

nge[nge.length - 1] = -1;

for (int i = arr.length - 2; i >= 0; i--) {

while (st.size() > 0 && arr[i] > st.peek()) {

st.pop();

}

if (st.size() == 0) {

nge[i] = -1;

} else {

nge[i] = st.peek();

}

st.push(arr[i]);

}

for (int val : nge) {

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

}

}

}