1. **An array contains 20 integers randomly. Write a java program to sort them in ascending order**

**Code:-**

/\*Write a program to input 20 random integers and sort using quick sort in ascending order

Author:Debayan De

\*/

// Java program for implementation of QuickSort

class QuickSort

{

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

{

int pivot = arr[high];

int i = (low-1); // index of smaller element

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

{

// If current element is smaller than or

// equal to pivot

if (arr[j] <= pivot)

{

i++;

// swap arr[i] and arr[j]

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

// swap arr[i+1] and arr[high] (or pivot)

int temp = arr[i+1];

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

arr[high] = temp;

return i+1;

}

/\* The main function that implements QuickSort()

arr[] --> Array to be sorted,

low --> Starting index,

high --> Ending index \*/

void sort(int arr[], int low, int high)

{

if (low < high)

{

/\* pi is partitioning index, arr[pi] is

now at right place \*/

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

// Recursively sort elements before

// partition and after partition

sort(arr, low, pi-1);

sort(arr, pi+1, high);

}

}

/\* A utility function to print array of size n \*/

static void printArray(int arr[])

{

int n = arr.length;

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

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

System.out.println();

}

// Driver program

public static void main(String args[])

{

int arr[] = {20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1};

int n = arr.length;

QuickSort ob = new QuickSort();

ob.sort(arr, 0, n-1);

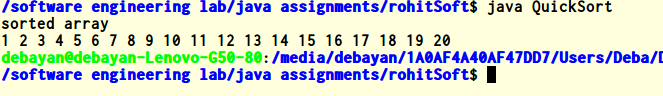
System.out.println("sorted array");

printArray(arr);

}

}

**Output:-**

****

**2.Write a java program that receives a integer no less than 20 and give factorial as output using recursive function call**

**Code:**

/\*Write a program to find the factorial of a number recursively

Author:- Debayan De

\*/

import java.util.Scanner;

class FactorialDemo{

public static void main(String args[]){

//Scanner object for capturing the user input

Scanner scanner = new Scanner(System.in);

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

//Stored the entered value in variable

int num = scanner.nextInt();

//Called the user defined function fact

int factorial = fact(num);

System.out.println("Factorial of entered number is: "+factorial);

}

static int fact(int n)

{

int output;

if(n==1){

return 1;

}

//Recursion: Function calling itself!!

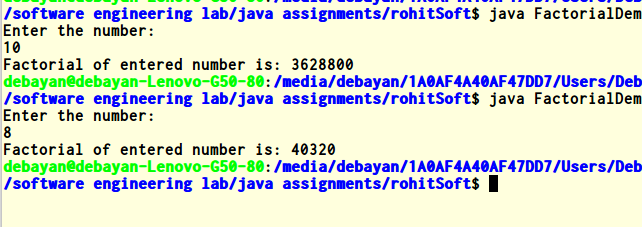
output = fact(n-1)\* n;

return output;

}

}

**Output:-**

****

**3.You are starting out on a long way.On the way there are N gas stations,the location of which are given as 1,2..N.Initially you are located at the gas station at 1 and your destination is N.Your car can run M station without refueling.Now you wish to plan your trip such that no of intermediate stop is minimum.**

**Input:-N and M.**

**Output:-The number of stop and the number of ways to trip**

**Code:-**

**Output:-**

**4.Given N separate integer points on the Cartesian plane satisfying there is no any other sharing same x coordinate.Count the no of rectangles created from any four of given points.**

**Code:-**

**Output:-**

**5.Given two stings,write a program that convert one string to other with minimum change**

**Code:-**

/\* A Naive recursive Java program to find minimum number

operations to convert str1 to str2

Author:-Debayan De

\*/

class EDIST

{

static int min(int x,int y,int z)

{

if (x<=y && x<=z) return x;

if (y<=x && y<=z) return y;

else return z;

}

static int editDist(String str1 , String str2 , int m ,int n)

{

// If first string is empty, the only option is to

// insert all characters of second string into first

if (m == 0) return n;

// If second string is empty, the only option is to

// remove all characters of first string

if (n == 0) return m;

// If last characters of two strings are same, nothing

// much to do. Ignore last characters and get count for

// remaining strings.

if (str1.charAt(m-1) == str2.charAt(n-1))

return editDist(str1, str2, m-1, n-1);

// If last characters are not same, consider all three

// operations on last character of first string, recursively

// compute minimum cost for all three operations and take

// minimum of three values.

return 1 + min ( editDist(str1, str2, m, n-1), // Insert

editDist(str1, str2, m-1, n), // Remove

editDist(str1, str2, m-1, n-1) // Replace

);

}

public static void main(String args[])

{

String str1 = "sunday";

String str2 = "saturday";

System.out.println( editDist( str1 , str2 , str1.length(), str2.length()) );

}

}

**Output:-**

**C:\Users\Deba\Desktop\software engineering lab\java assignments\5.png**

**6.A nxn magic square generation**

**Code:-**

// Java program to generate odd sized magic squares

//Author:-Debayan De

import java.io.\*;

class GFG

{

// Function to generate odd sized magic squares

static void generateSquare(int n)

{

int[][] magicSquare = new int[n][n];

// Initialize position for 1

int i = n/2;

int j = n-1;

// One by one put all values in magic square

for (int num=1; num <= n\*n; )

{

if (i==-1 && j==n) //3rd condition

{

j = n-2;

i = 0;

}

else

{

//1st condition helper if next number

// goes to out of square's right side

if (j == n)

j = 0;

//1st condition helper if next number is

// goes to out of square's upper side

if (i < 0)

i=n-1;

}

//2nd condition

if (magicSquare[i][j] != 0)

{

j -= 2;

i++;

continue;

}

else

//set number

magicSquare[i][j] = num++;

//1st condition

j++; i--;

}

// print magic square

System.out.println("The Magic Square for "+n+":");

System.out.println("Sum of each row or column "+n\*(n\*n+1)/2+":");

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

{

for(j=0; j<n; j++)

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

System.out.println();

}

}

// driver program

public static void main (String[] args)

{

// Works only when n is odd

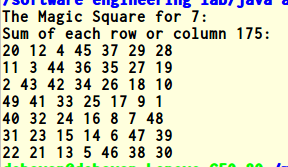
int n = 7;

generateSquare(n);

}

}

**Output:-**

****