Task 1: MATHEMATICS –WHY MATH IN CODING?

1a)

PROGRAM:

import java.util.Scanner;

public class Sqrt {

public static void main(String[] args) {

Scanner sc= new Scanner(System.in);

double a= sc.nextDouble();

double c= Math.sqrt(a);

long d= Math.round(c);

System.out.println(d);

}

}

1b)

PROGRAM:

import java.util.Scanner;

class UgluNum {

public static void main(String[] args){

Scanner sc= new Scanner(System.in);

int n= sc.nextInt();

if(Ugly(n)==1){

System.out.println(n+"is ugly ");

}

else{

System.out.println(n+"not ugly");

}

}

static int Ugly(int n){

if(n<=0){

return 0;

}

if(n%2==0){

n/=2;

}

if(n%3==0){

n/=3;

}

if(n%5==0){

n/=5;

}

return n;

}

}

1c)

PROGRAM:

import java.util.Scanner;

class ArrayMul {

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n= sc.nextInt();

int[] a= new int[n];

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

a[i]=sc.nextInt();

}

System.out.println(prod(a,n));

}

static int prod(int a[],int n){

int r=1;

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

r\*=a[i];

}

return r;

}

}

TASK 2: ADVANCED CONCEPT ON ARRAYS

2a)

PROGRAM:

Import java.util.\*;

public class IntervalList {

    public static int[][] intervalIntersection(int[][] firstList, int[][] secondList) {

        int i = 0, j = 0;

        List<int[]> result = new ArrayList<>();

        while (i < firstList.length && j < secondList.length) {

            int start = Math.max(firstList[i][0], secondList[j][0]);

            int end = Math.min(firstList[i][1], secondList[j][1]);

            if (start <= end) {

                result.add(new int[]{start, end});

            }

            if (firstList[i][1] < secondList[j][1]) {

                i++;

            } else {

                j++;

            }

        }

        return result.toArray(new int[result.size()][]);

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of intervals in first list: ");

        int n1 = sc.nextInt();

        int[][] firstList = new int[n1][2];

        System.out.println("Enter intervals for first list (start end):");

        for (int i = 0; i < n1; i++) {

            firstList[i][0] = sc.nextInt();

            firstList[i][1] = sc.nextInt();

        }

        System.out.print("Enter number of intervals in second list: ");

        int n2 = sc.nextInt();

        int[][] secondList = new int[n2][2];

        System.out.println("Enter intervals for second list :");

        for (int i = 0; i < n2; i++) {

            secondList[i][0] = sc.nextInt();

            secondList[i][1] = sc.nextInt();

        }

        int[][] intersections = intervalIntersection(firstList, secondList);

        System.out.println("Intersections:");

        for (int[] interval : intersections) {

            System.out.println("[" + interval[0] + ", " + interval[1] + "]");

        }

    }

}

2b)

PROGRAM:

import java.util.\*;

public class MergeSortedArrays {

public static int[] merge(int[] A, int[] B) {

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

int[] result = new int[A.length + B.length];

while (i < A.length && j < B.length) {

if (A[i] <= B[j])

result[k++] = A[i++];

else

result[k++] = B[j++];

}

while (i < A.length)

result[k++] = A[i++];

while (j < B.length)

result[k++] = B[j++];

return result;

}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in.nextInt();

int[] A = new int[n];

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

A[i] = in.nextInt();

int m= in.nextInt();

int[] B = new int[m];

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

B[i]=in.nextInt();

}

Arrays.sort(A);

Arrays.sort(B);

int[] merged = merge(A, B);

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

for (int num : merged) {

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

}

}

}

2C)

PROGRAM:

Import java.util.\*;

public class ThreeSum {

    public static List<List<Integer>> threeSum(int[] nums) {

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

        Arrays.sort(nums);

        for (int i = 0; i < nums.length - 2; i++) {

            if (i > 0 && nums[i] == nums[i - 1]) continue;

            int left = i + 1;

            int right = nums.length - 1;

            while (left < right) {

                int sum = nums[i] + nums[left] + nums[right];

                if (sum == 0) {

                    result.add(Arrays.asList(nums[i], nums[left], nums[right]));

                    while (left < right && nums[left] == nums[left + 1]) left++;

                    while (left < right && nums[right] == nums[right - 1]) right--;

                    left++;

                    right--;

                } else if (sum < 0) {

                    left++;

                } else {

                    right--;

                }

            }

        }

        return result;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int n = sc.nextInt();

        int[] nums = new int[n];

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

        for (int i = 0; i < n; i++) {

            nums[i] = sc.nextInt();

        }

        List<List<Integer>> triplets = threeSum(nums);

        System.out.println("Unique Triplets whose sum is 0:");

        for (List<Integer> triplet : triplets) {

            System.out.println(triplet);

        }

    }

}

TASK 3: ADVANCED CONCEPT ON STRINGS

3A) PROGRAM:

import java.util.\*;

public class Pattern {

public static void main(String[] args)

{

Scanner sc= new Scanner(System.in);

int n=sc.nextInt();

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

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

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

}

System.out.println();

}

}

}

3b)

import java.util.Scanner;

public class Palindrome {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a string or number: ");

String input = sc.nextLine();

String str = input.toLowerCase().replace(" ", "");

String reversed = "";

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

reversed += str.charAt(i);

}

if (str.equals(reversed)) {

System.out.println(input + " is a Palindrome.");

} else {

System.out.println(input + " is NOT a Palindrome.");

}

}

}

3c)

import java.util.Scanner;

public class PasswordValidator {

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter your password: ");

String password = sc.nextLine();

boolean isValid = *validatePassword*(password);

if (isValid) {

System.***out***.println("Password is valid.");

}

else {

System.***out***.println("Password is invalid. Please ensure it meets all criteria.");

}

}

public static boolean validatePassword(String password) {

boolean valid = true;

if (password.length() < 8) {

System.***out***.println("Password must be at least 8 characters long.");

valid = false;

}

if (!password.matches(".\*[A-Z].\*")) {

System.***out***.println("Password must contain at least one uppercase letter.");

valid = false;

}

if (!password.matches(".\*[a-z].\*")) {

System.***out***.println("Password must contain at least one lowercase letter.");

valid = false;

}

if (!password.matches(".\*[0-9].\*")) {

System.***out***.println("Password must contain at least one digit.");

valid = false;

}

if (!password.matches(".\*[@#$%^&+=!].\*")) {

System.out.println("Password must contain at least one special character (@, #, $, %, ^, &, +, =, !).");

valid = false;

}

return valid;

}

}

TASK 4:

4a)

import java.util.LinkedList;

import java.util.Queue;

public class StackUsingQueues {

private Queue<Integer> q1 = new LinkedList<>();

private Queue<Integer> q2 = new LinkedList<>();

public void push(int x) {

System.***out***.println("Pushing element: " + x);

q2.add(x);

while (!q1.isEmpty()) {

q2.add(q1.remove());

}

Queue<Integer> temp = q1;

q1 = q2;

q2 = temp;

}

public int pop() {

if (q1.isEmpty()) {

System.***out***.println("Stack is empty. Cannot pop.");

return -1;

}

int popped = q1.remove();

System.***out***.println("Popping element: " + popped);

return popped;

}

public int top() {

if (q1.isEmpty()) {

System.***out***.println("Stack is empty. No top element.");

return -1;

}

int top = q1.peek();

System.***out***.println("Top element: " + top);

return top;

}

public boolean isEmpty() {

boolean empty = q1.isEmpty();

System.***out***.println("Is stack empty? " + empty);

return empty;

}

public static void main(String[] args) {

StackUsingQueues stack = new StackUsingQueues();

stack.push(10);

stack.push(20);

stack.push(30);

stack.top();

stack.pop();

stack.top();

stack.isEmpty();

stack.pop();

stack.pop();

stack.isEmpty();

}

}

4b)

import java.util.Scanner;

public class PasswordValidator {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your password: ");

        String password = sc.nextLine();

        boolean isValid = validatePassword(password);

        if (isValid) {

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

        } else {

            System.out.println("Password is invalid. Please ensure it meets all criteria.");

        }

    }

    public static boolean validatePassword(String password) {

        boolean valid = true;

        if (password.length() < 8) {

            System.out.println("Password must be at least 8 characters long.");

            valid = false;

        }

        if (!password.matches(".\*[A-Z].\*")) {

            System.out.println("Password must contain at least one uppercase letter.");

            valid = false;

        }

        if (!password.matches(".\*[a-z].\*")) {

            System.out.println("Password must contain at least one lowercase letter.");

            valid = false;

        }

        if (!password.matches(".\*[0-9].\*")) {

            System.out.println("Password must contain at least one digit.");

            valid = false;

        }

        if (!password.matches(".\*[@#$%^&+=!].\*")) {

            System.out.println("Password must contain at least one special character (@, #, $, %, ^, &, +, =, !).");

            valid = false;

        }

        return valid;

    }

}

4c)

import java.util.PriorityQueue;

import java.util.Scanner;

import java.util.Collections;

public class Disk {

public static void solveTower(int[] disks, int day, int n, int max, PriorityQueue<Integer> pq) {

if (day > n) return; // Base case

System.***out***.print("Day " + day + ": ");

pq.add(disks[day - 1]);

while (!pq.isEmpty() && pq.peek() == max) {

System.***out***.print(pq.poll() + " ");

max--;

}

System.***out***.println();

*solveTower*(disks, day + 1, n, max, pq);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter the number of disks (days): ");

int n = sc.nextInt();

int[] disks = new int[n];

System.***out***.println("Enter the disk sizes:");

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

disks[i] = sc.nextInt();

}

System.***out***.println("Tower construction order:");

PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.*reverseOrder*());

*solveTower*(disks, 1, n, n, pq);

sc.close();

}

}