**PROGRAMMING IN JAVA WITH AWT**

**CSA – 0908**

**Shaik Nayaz Irfan Ali – 192211326**

**1.Write a Java program for Two sum.**

**PROGRAM**

import java.util.HashMap;

import java.util.Map;

public class R192211326 {

public static int[] findTwoSum(int[] nums, int target) {

Map<Integer, Integer> map = new HashMap<>();

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

int complement = target - nums[i];

if (map.containsKey(complement)) {

return new int[] { map.get(complement), i };

}

map.put(nums[i], i);

}

throw new IllegalArgumentException("No two sum solution");

}

public static void main(String[] args) {

int[] nums = {3,2,4};

int target = 6;

int[] result = findTwoSum(nums, target);

System.out.println("Indices: " + result[0] + ", " + result[1]);

}

}

**OUTPUT**

Indices: 1, 2

**2.Write a Java Program to check Palindrome or not.**

**PROGRAM**

public class R192211326{

public static void main(String args[]){

int x = 121;

int num = x;

int digit,remainder=0;

while(num!=0){

digit = num % 10;

remainder = remainder\*10 + digit;

num = num / 10;

}

if(remainder==x){

System.out.println(remainder + " is a palindrome");

}

else{

System.out.println(remainder + " is not a palindrome");

}

}

}

**OUTPUT**

121 is a Palindrome.

**3.Write a program to implement insertion sort on a array of integers using loops and conditional statements.**

**PROGRAM**

import java.util.Arrays;

class R192211326 {

public static void main(String[] args) {

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

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

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

j = j - 1;

}

arr[j + 1] = key;

}

System.out.println("Sorted array: " + Arrays.toString(arr));

}

}

**OUTPUT**

Sorted array: [2, 3, 4, 5, 6, 7, 8, 9]

**4.Merge Two Sorted Lists.**

**PROGRAM**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class R192211326 {

public static void main(String[] args) {

List<Integer> list1 = Arrays.asList(1, 3, 5, 7);

List<Integer> list2 = Arrays.asList(2, 4, 6, 8);

List<Integer> mergedList = mergeSortedLists(list1, list2);

System.out.println("Merged list: " + mergedList);

}

public static List<Integer> mergeSortedLists(List<Integer> list1, List<Integer> list2) {

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

int i = 0, j = 0;

while (i < list1.size() && j < list2.size()) {

if (list1.get(i) <= list2.get(j)) {

mergedList.add(list1.get(i));

i++;

} else {

mergedList.add(list2.get(j));

j++;

}

}

while (i < list1.size()) {

mergedList.add(list1.get(i));

i++;

}

while (j < list2.size()) {

mergedList.add(list2.get(j));

j++;

}

return mergedList;

}

}

**OUTPUT**

Merged list: [1, 2, 3, 4, 5, 6, 7, 8]

**5. Java program Roman to Integer.**

**PROGRAM**

public class R192211326 {

public static String intToRoman(int num) {

String[] romanSymbols = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};

int[] values = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};

StringBuilder roman = new StringBuilder();

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

while (num >= values[i]) {

roman.append(romanSymbols[i]);

num -= values[i];

}

}

return roman.toString();

}

public static void main(String[] args) {

int number = 1994;

String romanNumeral = intToRoman(number);

System.out.println("The Roman numeral for " + number + " is: " + romanNumeral);

}

}

**OUTPUT**

The Roman numeral for 1994 is: MCMXCIV

**6.Java Program for Longest Common Prefix.**

**PROGRAM**

public class R192211326 {

public static String longestCommonPrefix(String[] strs) {

if (strs == null || strs.length == 0) {

return "";

}

String prefix = strs[0];

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

while (strs[i].indexOf(prefix) != 0) {

prefix = prefix.substring(0, prefix.length() - 1);

if (prefix.isEmpty()) {

return "";

}

}

}

return prefix;

}

public static void main(String[] args) {

String[] strs = {"flower", "flow", "flight"};

String result = longestCommonPrefix(strs);

System.out.println("The longest common prefix is: " + result);

}

}

**OUTPUT**

The longest common prefix is: f l

**7.Java Program to ADD Binary.**

**PROGRAM**

public class R192211326 {

public static String addBinary(String a, String b) {

int sum = 0;

int i = a.length() - 1;

int j = b.length() - 1;

int carry = 0;

StringBuilder result = new StringBuilder();

while (i >= 0 || j >= 0 || carry != 0) {

sum = carry;

if (i >= 0) {

sum += a.charAt(i--) - '0';

}

if (j >= 0) {

sum += b.charAt(j--) - '0';

}

carry = sum / 2;

result.append(sum % 2);

}

return result.reverse().toString();

}

public static void main(String[] args) {

String a = "11";

String b = "11";

String result = addBinary(a, b);

System.out.println("The sum of " + a + " and " + b + " is " + result);

}

}

**OUTPUT**

The sum of 11 and 11 is 110

**8.Java Program For Square Root.**

**PROGRAM**

public class R192211326 {

public static void main(String[] args) {

int x = 8;

int sqrt = sqrtRounded(x);

System.out.println("The square root of " + x + " rounded to the nearest integer is " + sqrt);

}

public static int sqrtRounded(int x) {

if (x < 0) {

throw new ArithmeticException("Square root of negative number");

}

if (x == 0 || x == 1) {

return x;

}

int start = 1;

int end = x;

int result = 0;

while (start <= end) {

int mid = start + (end - start) / 2;

long square = (long) mid \* mid;

if (square == x) {

return mid;

} else if (square < x) {

start = mid + 1;

result = mid;

} else {

end = mid - 1;

}

}

return result;

}

}

**OUTPUT**

The square root of 8 rounded to the nearest integer is 2.

**9.Java Program For Single Number.**

import java.util.\*;

public class R192211326{

public static int findSingleNumber(int[] nums) {

Map<Integer, Integer> countMap = new HashMap<>();

for (int num : nums) {

countMap.put(num, countMap.getOrDefault(num, 0) + 1);

}

for (Map.Entry<Integer, Integer> entry : countMap.entrySet()) {

if (entry.getValue() == 1) {

return entry.getKey();

}

}

throw new RuntimeException("No single number found");

}

public static void main(String[] args) {

int[] nums = {2, 2, 1};

int singleNumber = findSingleNumber(nums);

System.out.println("Single number: " + singleNumber);

}

}

**OUTPUT**

Single number: 1

**10.Java Program For Arthematic Operators**

**PROGRAM**

public class R192211326 {

public static void main(String[] args) {

int num1 = 10; // You can change this value

int num2 = 5; // You can change this value

int sum = num1 + num2;

int difference = num1 - num2;

int product = num1 \* num2;

double quotient = (double) num1 / num2;

System.out.println("For the numbers " + num1 + " and " + num2 + ":");

System.out.println("Sum: " + sum);

System.out.println("Difference: " + difference);

System.out.println("Product: " + product);

System.out.println("Quotient: " + quotient);

}

}

**OUTPUT**

For the numbers 10 and 5:

Sum: 15

Difference: 5

Product: 50

Quotient: 2.0

**11.Java Program For Linked List.**

**PROGRAM**

public class R192211326 {

public static boolean hasCycle(Node head) {

if (head == null) return false;

Node slow = head;

Node fast = head;

while (fast != null && fast.next != null) {

slow = slow.next;

fast = fast.next.next;

if (slow == fast) {

return true;

}

}

return false;

}

public static void main(String[] args) {

Node node1 = new Node(1);

Node node2 = new Node(2);

Node node3 = new Node(3);

Node node4 = new Node(4);

node1.next = node2;

node2.next = node3;

node3.next = node4;

node4.next = node2; // create a cycle

boolean hasCycle = hasCycle(node1);

System.out.println("Has cycle: " + hasCycle);

}

}

class Node {

int value;

Node next;

public Node(int value) {

this.value = value;

this.next = null;

}

}

**OUTPUT**

Has cycle: true

**12.Java Program For**

**PROGRAM**

public class R192211326 {

public static void main(String[] args) {

int number = 28;

boolean isPerfect = isPerfectNumber(number);

if (isPerfect) {

System.out.println(number + " is a perfect number.");

} else {

System.out.println(number + " is not a perfect number.");

}

}

public static boolean isPerfectNumber(int number) {

if (number <= 1) {

return false;

}

int sumOfDivisors = 0;

for (int i = 1; i <= number / 2; i++) {

if (number % i == 0) {

sumOfDivisors += i;

}

}

return sumOfDivisors == number;

}

}

**OUTPUT**

28 is a perfect number.

**13.Java program for Product of given number.**

**PROGRAM**

public class R192211326 {

public static void main(String[] args) {

int number = 1234;

int product = calculateProductOfDigits(number);

System.out.println("The product of the digits of " + number + " is: " + product);

}

public static int calculateProductOfDigits(int number) {

number = Math.abs(number);

if (number == 0) {

return 0;

}

int product = 1;

while (number > 0) {

int digit = number % 10;

product \*= digit;

number /= 10;

}

return product;

}

}

**OUTPUT**

The product of the digits of 1234 is: 24