**Coding practice Problems:** 12.11.2024  
  **Name: SUJITHA M  
  Dept :CSE**

1. **Anagram program**

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String str1 = scanner.nextLine();

String str2 = scanner.nextLine();

if (areAnagrams(str1, str2)) {

System.out.println("Anagram");

} else {

System.out.println("Not Anagram");

}

}

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

if (str1.length() != str2.length()) {

return false;

}

char[] arr1 = str1.toCharArray();

char[] arr2 = str2.toCharArray();

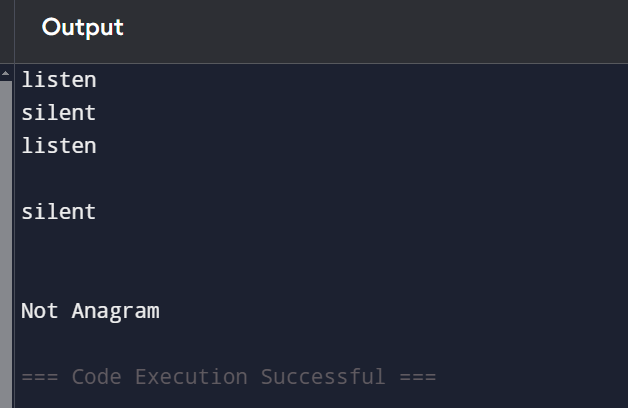
java.util.Arrays.sort(arr1);

java.util.Arrays.sort(arr2);

return java.util.Arrays.equals(arr1, arr2);

}

}



2. Row with Max One’s

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int m = scanner.nextInt();

int n = scanner.nextInt();

int[][] matrix = new int[m][n];

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

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

matrix[i][j] = scanner.nextInt();

}

}

int maxRow = -1, maxCount = -1;

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

int count = 0;

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

if (matrix[i][j] == 1) {

count++;

}

}

if (count > maxCount) {

maxCount = count;

maxRow = i;

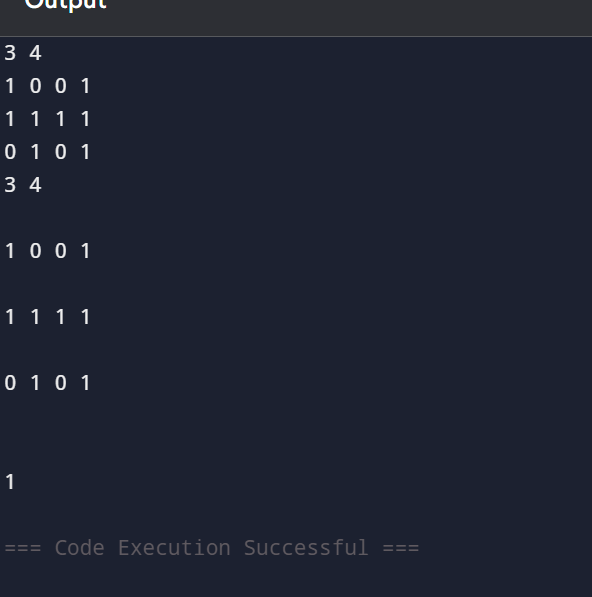
}

}

System.out.println(maxRow);

}

}



3. Longest consequtive subsequence

import java.util.Scanner;

import java.util.HashSet;

class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

int[] arr = new int[n];

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

arr[i] = scanner.nextInt();

}

System.out.println(longestConsecutiveSubsequence(arr));

}

public static int longestConsecutiveSubsequence(int[] arr) {

HashSet<Integer> set = new HashSet<>();

for (int num : arr) {

set.add(num);

}

int longestStreak = 0;

for (int num : arr) {

if (!set.contains(num - 1)) {

int currentNum = num;

int currentStreak = 1;

while (set.contains(currentNum + 1)) {

currentNum++;

currentStreak++;

}

longestStreak = Math.max(longestStreak, currentStreak);

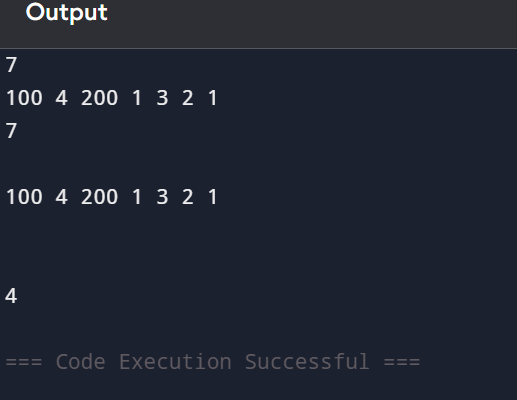
}

}

return longestStreak;

}

}



4. Longest Palindromic Substring

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String s = scanner.nextLine();

System.out.println(longestPalindromicSubstring(s));

}

public static String longestPalindromicSubstring(String s) {

if (s == null || s.length() < 1) {

return "";

}

int start = 0, maxLength = 1;

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

int len1 = expandAroundCenter(s, i, i);

int len2 = expandAroundCenter(s, i, i + 1);

int len = Math.max(len1, len2);

if (len > maxLength) {

maxLength = len;

start = i - (len - 1) / 2;

}

}

return s.substring(start, start + maxLength);

}

public static int expandAroundCenter(String s, int left, int right) {

while (left >= 0 && right < s.length() && s.charAt(left) == s.charAt(right)) {

left--;

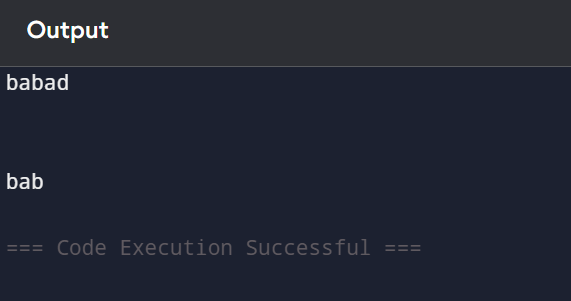
right++;

}

return right - left - 1;

}

}



5. Rat in a Maze

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

int[][] maze = new int[n][n];

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

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

maze[i][j] = scanner.nextInt();

}

}

int[][] path = new int[n][n];

if (ratInMaze(maze, 0, 0, path)) {

printSolution(path);

} else {

System.out.println("No solution");

}

}

public static boolean ratInMaze(int[][] maze, int x, int y, int[][] path) {

int n = maze.length;

if (x == n - 1 && y == n - 1 && maze[x][y] == 1) {

path[x][y] = 1;

return true;

}

if (isSafe(maze, x, y)) {

path[x][y] = 1;

if (ratInMaze(maze, x + 1, y, path)) {

return true;

}

if (ratInMaze(maze, x, y + 1, path)) {

return true;

}

path[x][y] = 0;

return false;

}

return false;

}

public static boolean isSafe(int[][] maze, int x, int y) {

int n = maze.length;

return x >= 0 && x < n && y >= 0 && y < n && maze[x][y] == 1;

}

public static void printSolution(int[][] path) {

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

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

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

}

System.out.println();

}

}

}

