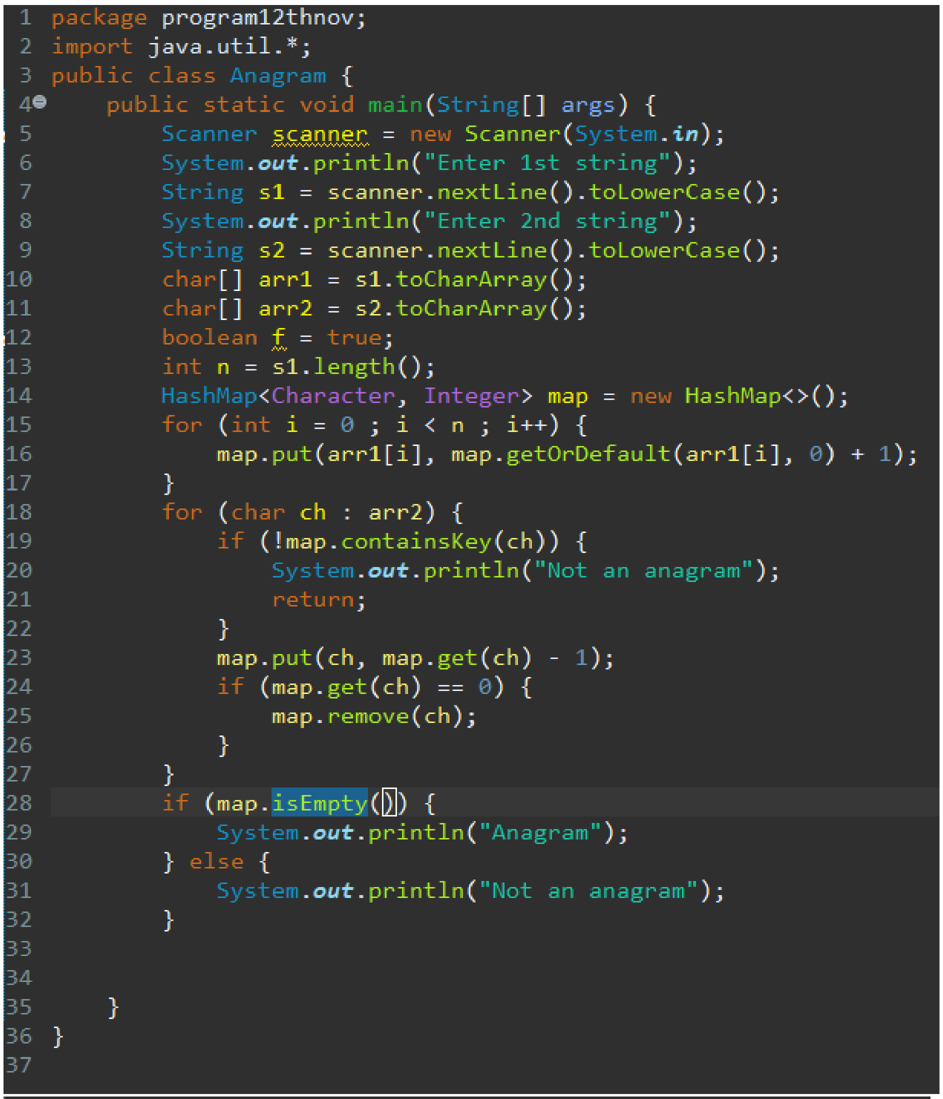
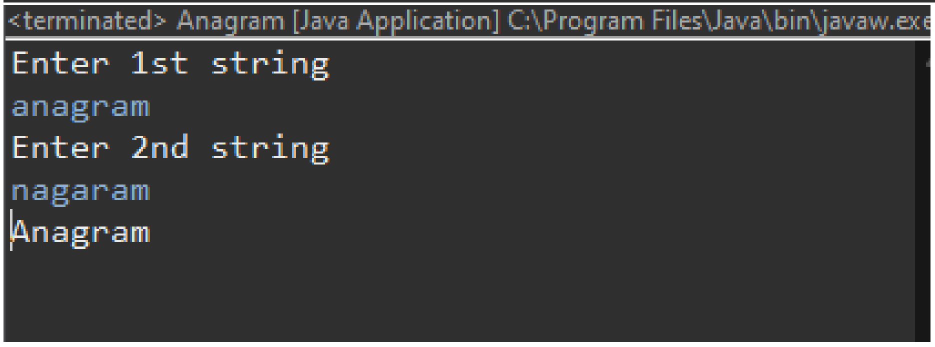
Problem Solving - 12/11/2024

1. Anagram Program:
   Time Complexity: O(n)
   Space Complexity: O(n)



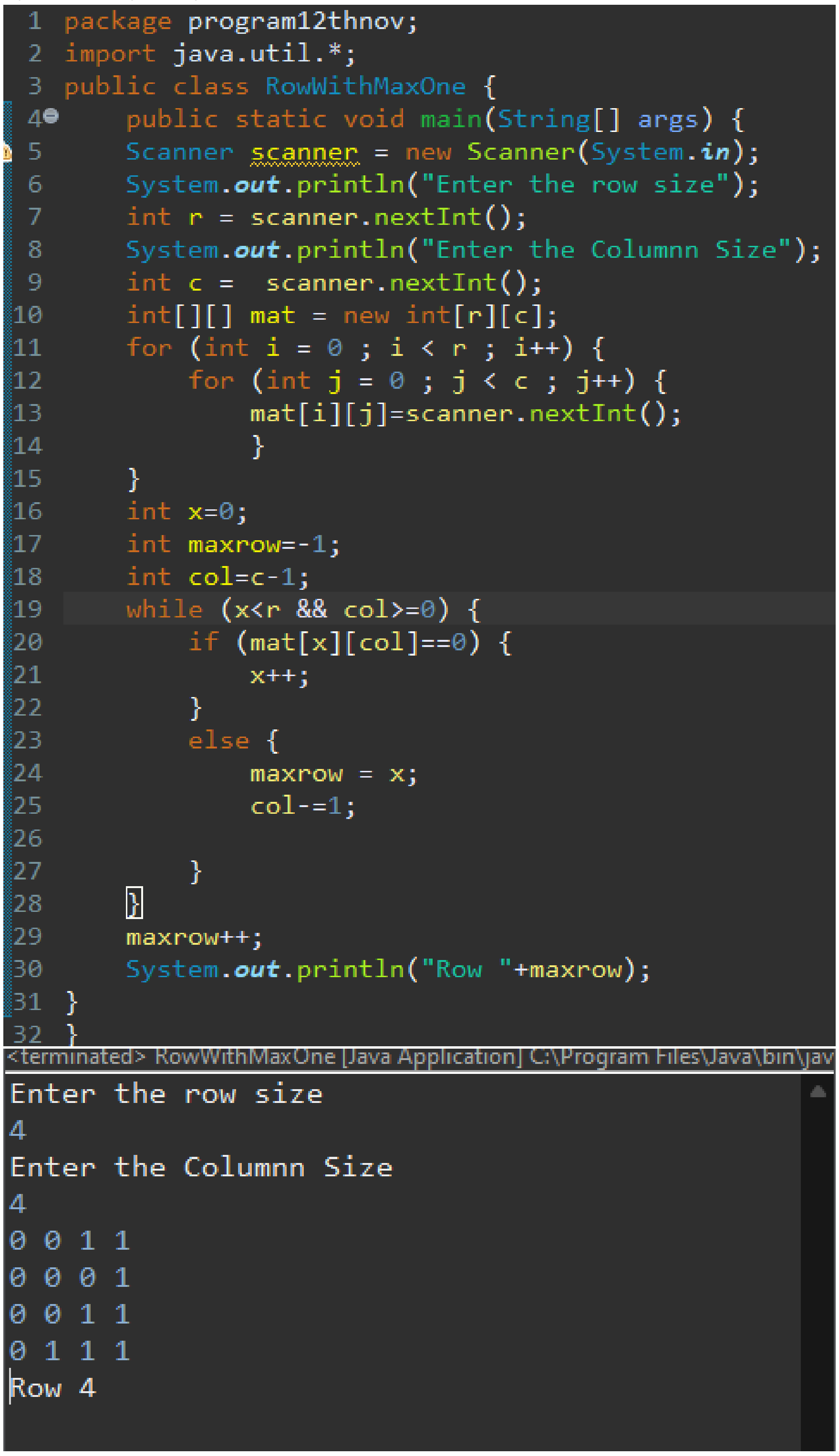
```java
package program12thnov;
import java.util.*;
public class Anagram {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter 1st string");
        String s1 = scanner.nextLine().toLowerCase();
        System.out.println("Enter 2nd string");
        String s2 = scanner.nextLine().toLowerCase();
        char[] arr1 = s1.toCharArray();
        char[] arr2 = s2.toCharArray();
        boolean f = true;
        int n = s1.length();
        HashMap<Character, Integer> map = new HashMap<>();
        for (int i = 0 ; i < n ; i++) {
            map.put(arr1[i], map.getOrDefault(arr1[i], 0) + 1);
        }
        for (char ch : arr2) {
            if (!map.containsKey(ch)) {
                System.out.println("Not an anagram");
                return;
            }
            map.put(ch, map.get(ch) - 1);
            if (map.get(ch) == 0) {
                map.remove(ch);
            }
        }
        if (map.isEmpty()) {
            System.out.println("Anagram");
        } else {
            System.out.println("Not an anagram");
        }


    }
}
```



```
<terminated> Anagram [Java Application] C:\Program Files\Java\bin\javaw.exe
Enter 1st string
anagram
Enter 2nd string
nagaram
Anagram
```

2. Row With Maximum One
   Time Complexity: O(r)
   Space Complexity: O(r * c)

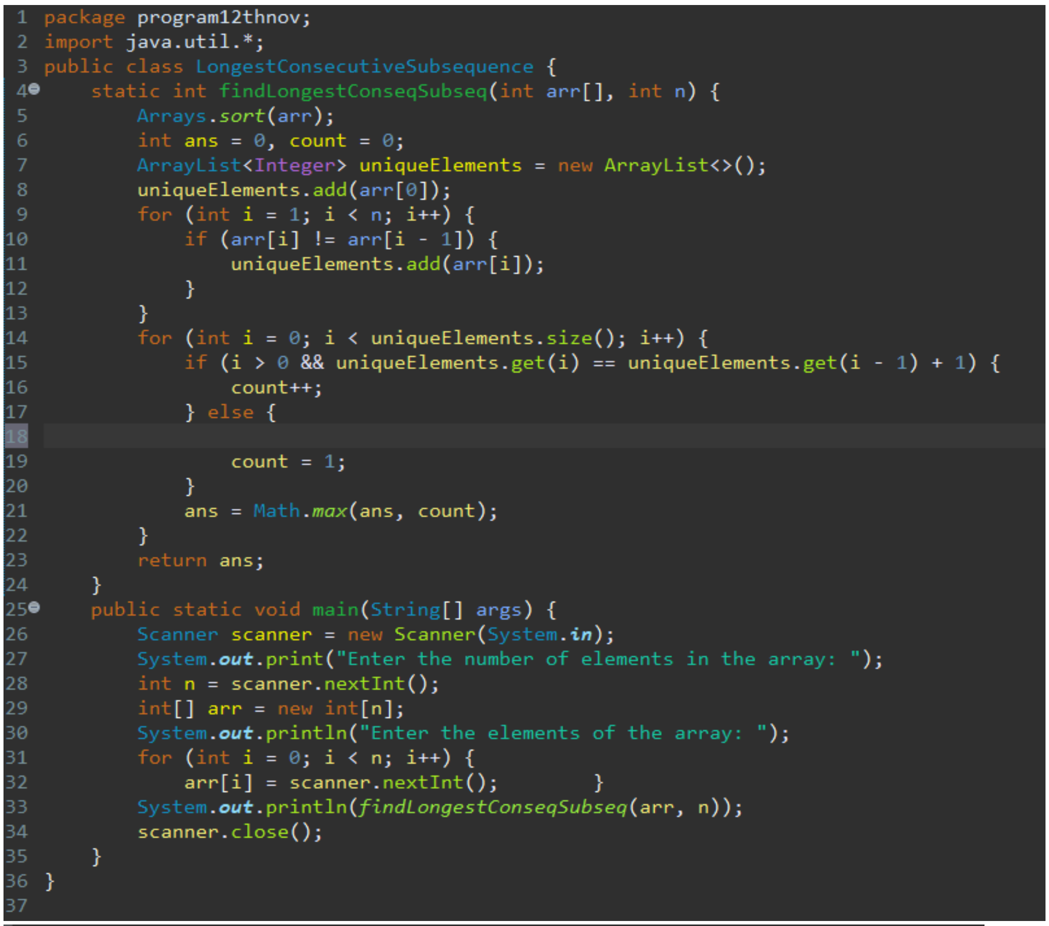


```java
package program12thnov;
import java.util.*;
public class RowWithMaxOne {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the row size");
    int r = scanner.nextInt();
    System.out.println("Enter the Columnn Size");
    int c =  scanner.nextInt();
    int[][] mat = new int[r][c];
    for (int i = 0 ; i < r ; i++) {
        for (int j = 0 ; j < c ; j++) {
            mat[i][j]=scanner.nextInt();
            }
    }
    int x=0;
    int maxrow=-1;
    int col=c-1;
    while (x<r && col>=0) {
        if (mat[x][col]==0) {
            x++;
        }
        else {
            maxrow = x;
            col-=1;

        }
    }
    maxrow++;
    System.out.println("Row "+maxrow);
}
}
```

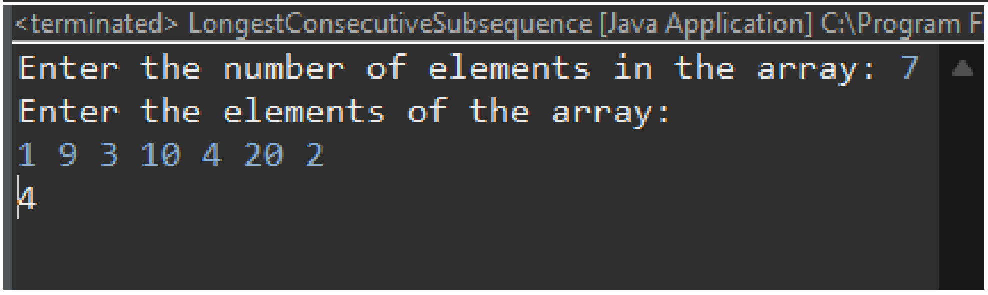
```
<terminated> RowWithMaxOne [Java Application] C:\Program Files\Java\bin\jav
Enter the row size
4
Enter the Columnn Size
4
0 0 1 1
0 0 0 1
0 0 1 1
0 1 1 1
Row 4
```

3. Longest Consecutive Subsequence
   Time Complexity: O(n log n)
   Space Complexity: O(n)



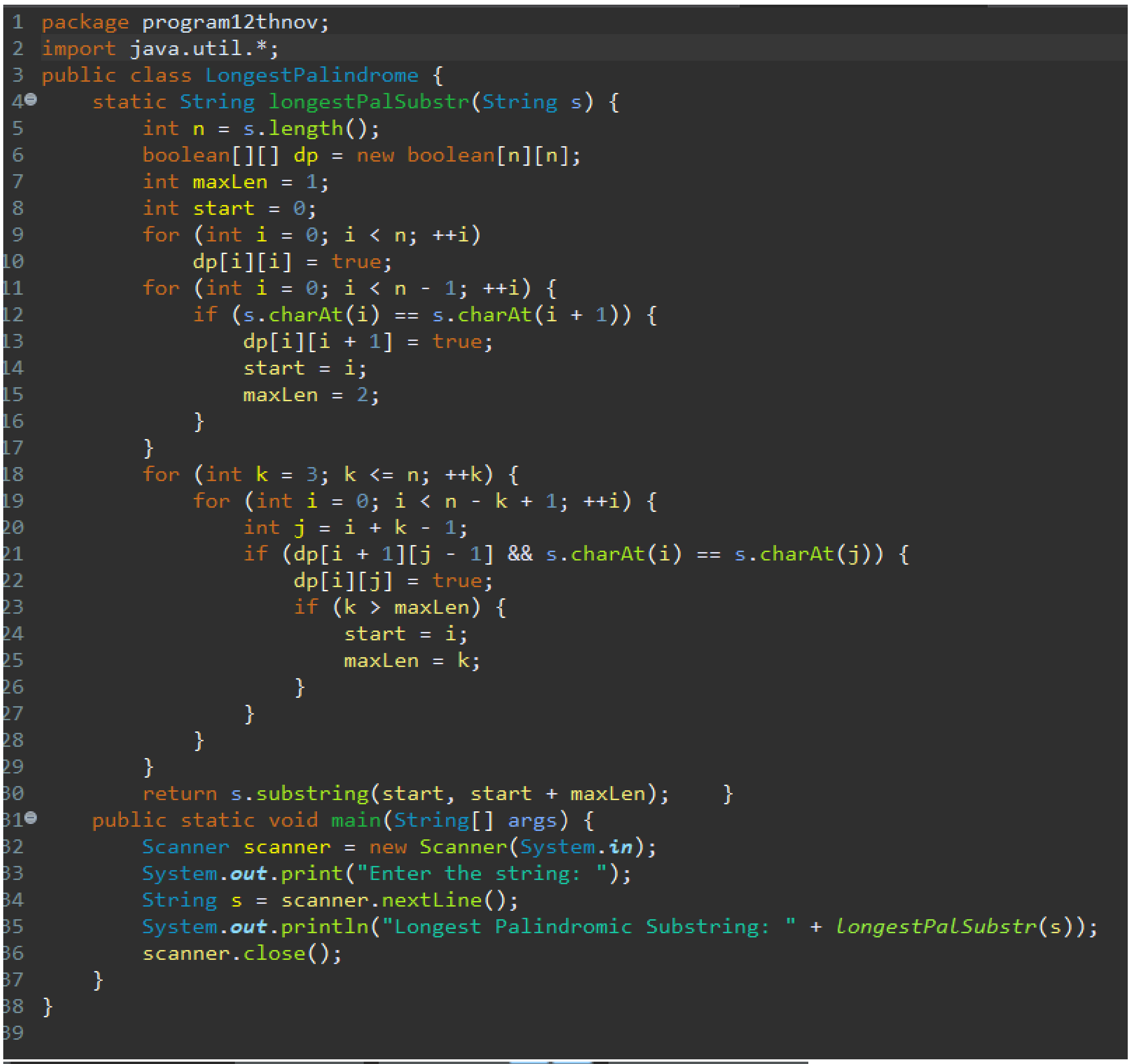
```java
package program12thnov;
import java.util.*;
public class LongestConsecutiveSubsequence {
    static int findLongestConseqSubseq(int arr[], int n) {
        Arrays.sort(arr);
        int ans = 0, count = 0;
        ArrayList<Integer> uniqueElements = new ArrayList<>();
        uniqueElements.add(arr[0]);
        for (int i = 1; i < n; i++) {
            if (arr[i] != arr[i - 1]) {
                uniqueElements.add(arr[i]);
            }
        }
        for (int i = 0; i < uniqueElements.size(); i++) {
            if (i > 0 && uniqueElements.get(i) == uniqueElements.get(i - 1) + 1) {
                count++;
            } else {

                count = 1;
            }
            ans = Math.max(ans, count);
        }
        return ans;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();        }
        System.out.println(findLongestConseqSubseq(arr, n));
        scanner.close();
    }
}
```

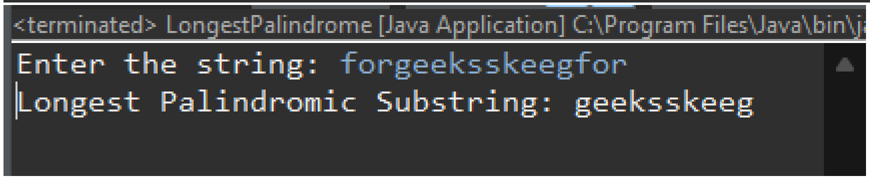


```
<terminated> LongestConsecutiveSubsequence [Java Application] C:\Program F
Enter the number of elements in the array: 7
Enter the elements of the array:
1 9 3 10 4 20 2
4
```

4. Longest Palindrome String
   Time Complexity: O(n^2)
   Space Complexity: O(n^2)



```java
package program12thnov;
import java.util.*;
public class LongestPalindrome {
    static String longestPalSubstr(String s) {
        int n = s.length();
        boolean[][] dp = new boolean[n][n];
        int maxLen = 1;
        int start = 0;
        for (int i = 0; i < n; ++i)
            dp[i][i] = true;
        for (int i = 0; i < n - 1; ++i) {
            if (s.charAt(i) == s.charAt(i + 1)) {
                dp[i][i + 1] = true;
                start = i;
                maxLen = 2;
            }
        }
        for (int k = 3; k <= n; ++k) {
            for (int i = 0; i < n - k + 1; ++i) {
                int j = i + k - 1;
                if (dp[i + 1][j - 1] && s.charAt(i) == s.charAt(j)) {
                    dp[i][j] = true;
                    if (k > maxLen) {
                        start = i;
                        maxLen = k;
                    }
                }
            }
        }
        return s.substring(start, start + maxLen);    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the string: ");
        String s = scanner.nextLine();
        System.out.println("Longest Palindromic Substring: " + longestPalSubstr(s));
        scanner.close();
    }
}
```

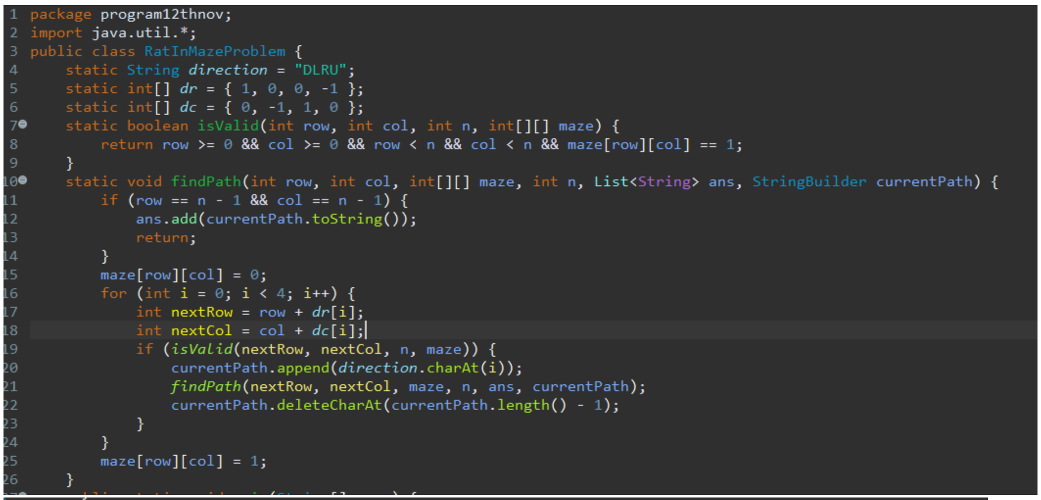


```
<terminated> LongestPalindrome [Java Application] C:\Program Files\Java\bin\j
Enter the string: forgeeksskeegfor
Longest Palindromic Substring: geeksskeeg
```

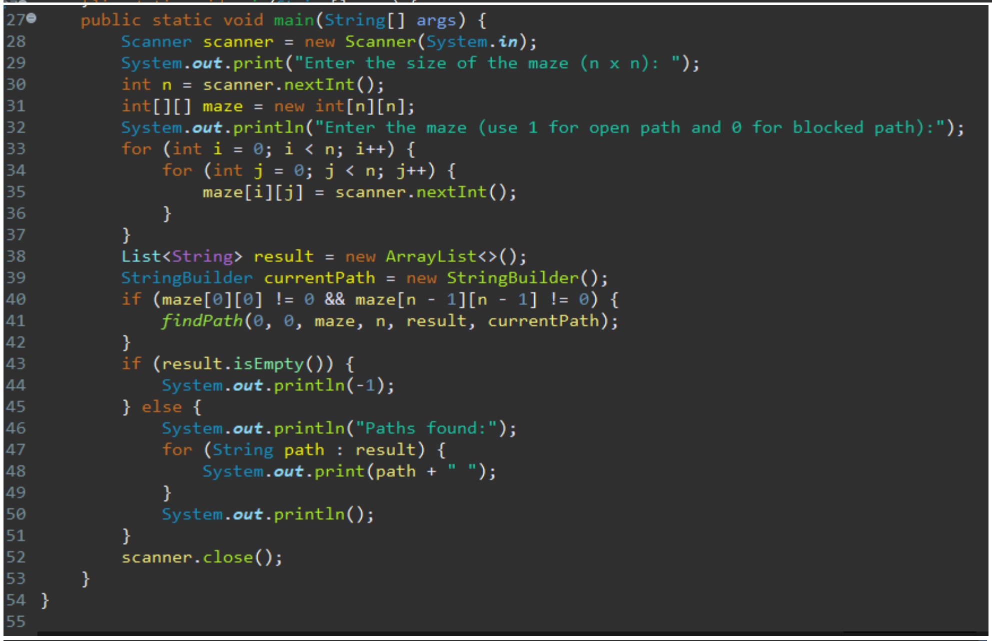
5. Rat in a Maze Problem

Time Complexity: O(4^(n^2))

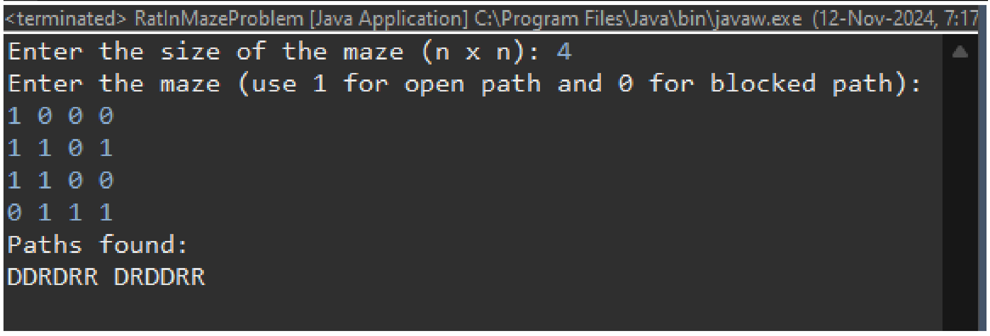
Space Complexity: O(n^2)



```java
package program12thnov;
import java.util.*;
public class RatInMazeProblem {
    static String direction = "DLRU";
    static int[] dr = { 1, 0, 0, -1 };
    static int[] dc = { 0, -1, 1, 0 };
    static boolean isValid(int row, int col, int n, int[][] maze) {
        return row >= 0 && col >= 0 && row < n && col < n && maze[row][col] == 1;
    }
    static void findPath(int row, int col, int[][] maze, int n, List<String> ans, StringBuilder currentPath) {
        if (row == n - 1 && col == n - 1) {
            ans.add(currentPath.toString());
            return;
        }
        maze[row][col] = 0;
        for (int i = 0; i < 4; i++) {
            int nextRow = row + dr[i];
            int nextCol = col + dc[i];
            if (isValid(nextRow, nextCol, n, maze)) {
                currentPath.append(direction.charAt(i));
                findPath(nextRow, nextCol, maze, n, ans, currentPath);
                currentPath.deleteCharAt(currentPath.length() - 1);
            }
        }
        maze[row][col] = 1;
    }
```



```java
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the maze (n x n): ");
        int n = scanner.nextInt();
        int[][] maze = new int[n][n];
        System.out.println("Enter the maze (use 1 for open path and 0 for blocked path):");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                maze[i][j] = scanner.nextInt();
            }
        }
        List<String> result = new ArrayList<>();
        StringBuilder currentPath = new StringBuilder();
        if (maze[0][0] != 0 && maze[n - 1][n - 1] != 0) {
            findPath(0, 0, maze, n, result, currentPath);
        }
        if (result.isEmpty()) {
            System.out.println(-1);
        } else {
            System.out.println("Paths found:");
            for (String path : result) {
                System.out.print(path + " ");
            }
            System.out.println();
        }
        scanner.close();
    }
}
```



```
<terminated> RatInMazeProblem [Java Application] C:\Program Files\Java\bin\javaw.exe (12-Nov-2024, 7:17
Enter the size of the maze (n x n): 4
Enter the maze (use 1 for open path and 0 for blocked path):
1 0 0 0
1 1 0 1
1 1 0 0
0 1 1 1
Paths found:
DDRDRR DRDDRR
```