1. Number of Longest Increasing Subsequence

Program:

```
import java.util.Arrays;
public class Solution {
    public int findNumberOfLIS(int[] nums) {
        int n = nums.length, maxLen = 0, ans = 0;
        int[] length = new int[n], count = new int[n];
        Arrays.fill(length, 1);
        Arrays.fill(count, 1);
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < i; ++j) {
                if (nums[j] < nums[i]) {
                    if (length[j] + 1 > length[i]) {
                        length[i] = length[j] + 1;
                        count[i] = count[j];
                    } else if (length[j] + 1 == length[i]) {
                        count[i] += count[j];
                    }
                }
            }
            if (length[i] > maxLen) {
                maxLen = length[i];
                ans = count[i];
            } else if (length[i] == maxLen) {
                ans += count[i];
            }
        }
        return ans;
    }
    public static void main(String[] args) {
        Solution sol = new Solution();
        int[] nums1 = {1, 3, 5, 4, 7};
        System.out.println("Input: [1, 3, 5, 4, 7]");
        System.out.println("Output: " + sol.findNumberOfLIS(nums1));

        int[] nums2 = {2, 2, 2, 2, 2};
        System.out.println("Input: [2, 2, 2, 2, 2]");
        System.out.println("Output: " + sol.findNumberOfLIS(nums2));
    }
}
```

OUTPUT:

```
2
5
```

--

2 . Wildcard Matching

Program:

```
public class WildcardMatching {
    public boolean isMatch(String s, String p) {
        int m = s.length(), n = p.length();
        boolean[][] dp = new boolean[m + 1][n + 1];
        dp[0][0] = true;

        for (int j = 1; j <= n; j++)
            if (p.charAt(j - 1) == '*')
                dp[0][j] = dp[0][j - 1];

        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                char sc = s.charAt(i - 1), pc = p.charAt(j - 1);
                if (pc == '*')
                    dp[i][j] = dp[i - 1][j] || dp[i][j - 1];
                else if (pc == '?' || sc == pc)
                    dp[i][j] = dp[i - 1][j - 1];
            }
        }
        return dp[m][n];
    }

    public static void main(String[] args) {
        WildcardMatching w = new WildcardMatching();
        System.out.println(w.isMatch("adceb", "*a*b"));
        System.out.println(w.isMatch("acdcb", "a*c?b"));
    }
}

}
```

OUTPUT:

```
 true
false
```