**Topic:** Recursion

**Topic Introduction:**

The topic of this lab is recursion, which is a programming technique where a function calls itself to solve a problem. Recursion can be used to solve many problems that involve repeating patterns, such as searching through a tree data structure or computing a mathematical sequence. In this lab, we will explore a creative problem that can be solved using recursion.

**Assignment Goals:**

* Understand the concept of recursion and how it can be used to solve problems
* Design and implement a recursive function to solve a creative problem
* Analyze the time and space complexity of a recursive solution

**Assignment Description:**

Imagine you are given a list of strings, where each string represents a possible combination of parentheses of varying length. For example, you might have a list like this:

["()", "(())", "()()", "((()))", "(()())", "()(())", "(((())))", "()()()"]

Your task is to write a function that returns the length of the longest valid sequence of parentheses in the list. A valid sequence of parentheses is one where each open parenthesis has a corresponding closing parenthesis, and no closing parenthesis is used without a corresponding open parenthesis. For example, "(())" and "()()" are valid sequences, but ")()(" and "((())" are not.

**Starting code:**

|  |
| --- |
| public class Parentheses {      public static boolean isValidParentheses(String s, int numOpen) {          // Your code here      }        public static int longestValidSequence(String[] strings) {          // Your code here          return 0;      }        public static void main(String[] args) {          String[] testStrings = {"()", "(())", "()()", "((()))", "(()())", "()(())", "(((())))", "()()()"};          System.out.println(longestValidSequence(testStrings)); // Expected output: 4      }  } |

**Key:**

|  |
| --- |
| public class Parentheses {      public static boolean isValidParentheses(String s, int numOpen) {          if (s.equals("") && numOpen == 0) {              return true;          }          if (s.equals("") || numOpen < 0) {              return false;          }          if (s.charAt(0) == '(') {              return isValidParentheses(s.substring(1), numOpen + 1);          } else if (s.charAt(0) == ')' && numOpen > 0) {              return isValidParentheses(s.substring(1), numOpen - 1);          } else {              return isValidParentheses(s.substring(1), numOpen);          }      }        public static int longestValidSequence(String[] strings) {          int longest = 0;          for (String s : strings) {              int length = 0;              for (int i = 0; i < s.length(); i++) {                  if (isValidParentheses(s.substring(i), 0)) {                      length++;                  }              }              if (length > longest) {                  longest = length;              }          }          return longest/2;      }        public static void main(String[] args) {          String[] testStrings = {"()", "(())", "()()", "((()))", "(()())", "()(())", "(((())))", "()()()"};          System.out.println(longestValidSequence(testStrings)); // Expected output: 4      }  } |