COP3503

Assignment 2

Assignment Objective:

Since backtracking was an important topic we covered, this assignment should help deepen our understanding of how to perform backtracking

Assignment Description

You have been given a task to **generate all valid combinations** of **n pairs** of parentheses. Each combination must be a **well-formed expression**, meaning:

- Every opening parenthesis `(` must have a corresponding closing parenthesis `)`
- No closing parenthesis `)` can come before an unmatched opening parenthesis `(`

For example, given `n = 3`, the possible valid combinations of parentheses are:

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

Your goal is to implement a method that efficiently finds all valid well-formed parentheses combinations using **backtracking**.

Implementation Details

1. You will create a class called **ParenthesesCombinations**.
2. There is no need to implement an explicit constructor; the default constructor will be used.
3. Implement a non-static method called generateParentheses that takes one parameter:
    - An integer `n`, representing the number of pairs of parentheses.
    - The method must return a List of Strings, where each string represents a valid parentheses combination.
4. You must use **backtracking** to generate all possible solutions efficiently.
5. You can create helper methods as needed, but they should be implemented within the same file.

Constraints and Assumptions

- `1 ≤ n ≤ 20`
- The output list should contain all possible valid combinations.
- The output order does not matter.

About the Driver File

A driver file (`ParenthesesCombinationsDriver.java`) will be provided to demonstrate how the `generateParentheses` method is called, along with 5 test cases to validate your implementation.

DO NOT MODIFY THE DRIVER FILE
The graders will use a new driver file with some different test cases, so ensure your code works with the provided driver file.

What to Submit

Submit **one file**: `ParenthesesCombinations.java`

You do not need to submit the driver file. Ensure your program runs successfully using the provided driver file. Incorrect naming or missing files will result in a lower grade.

Important Notes for Running in Eustis

- If you are using an IDE like NetBeans or Eclipse, ensure your Java file is not inside a package.
- Do NOT include a `main` method in your solution file. The driver function will handle the call to your function
- Any compilation errors due to package naming will NOT be fixed during grading and will result in point deductions.

Code Style Requirements

- Please follow the provided code style guidelines (available on WebCourses).