[CS 11 25.1] HOPE 2h – Fancy Calculator

Cheatsheet is available here: https://oj.dcs.upd.edu.ph/cs11cheatsheet/

Submit solution [CS 11 25.1] HOPE 2

✓ **Points:** 170 (partial)
Time limit: 3.0s
Memory limit: 2G

› **Problem type**
⌄ **Allowed languages**
py3

Problem Statement

Yuuka wants to make her calculator even fancier, so she has come up with a new feature: *pair visualization*.

Suppose we have the expression `(a * b) + (c * d)`. Here, the parenthesis before the `a` pairs up with the parenthesis after the `b`, and the parenthesis before the `c` pairs up with the parenthesis after the `d`.

Visually, these pairs can be shown like this:

```
╭─────╮   ╭─────╮
(a * b) + (c * d)
```

Given an expression, can you visualize its pairs?

Here are the special characters that you will need to solve this problem:

```
╭ ─ ╮ │
```

Note that the `│` character here is not the same as the vertical pipe character in your keyboard!

Important: Each "line" visualizing a pair should be as short as possible.

Task Details

Your task is to implement a function named `visualize_pairs`, which should start like this:

```
def visualize_pairs(expr):
```

Here, `expr` is a string representing the given expression.

The function must return a tuple of strings, where each string corresponds to a row of the visualization.

Restrictions

- Loops and lists are allowed.
- Sets and dictionaries are allowed.
- Generators and comprehensions are allowed.
- Recursion is allowed.
- Your source code must have at most 2500 bytes.

Examples

Example 1 Function Call

```
visualize_pairs("(a * b) + (c * d)")
```

Example 1 Return Value

```
(
    "╭─────╮   ╭─────╮",
    "(a * b) + (c * d)",
)
```

Example 2 Function Call

```
visualize_pairs("(x * (a + b + c)) - 100 / (1 + [[y]])")
```

Example 2 Return Value

```
(
    "                          ╭─────────╮",
    "╭───────────────╮         │    ╭───╮│",
    "│    ╭─────────╮│         │    │╭─╮││",
    "(x * (a + b + c)) - 100 / (1 + [[y]])",
)
```

Example 3 Function Call

```
visualize_pairs("((1 + (2)) + (3))")
```

Example 3 Return Value

```
(
    '╭───────────────╮',
    '│╭───────╮      │',
    '││    ╭─╮│   ╭─╮│',
    '((1 + (2)) + (3))',
)
```

Constraints

Let ℓ be the length of the expression in a single call to `visualize_pairs`.

- The function `visualize_pairs` will be called at most 50 times.
- $1 \leq \ell \leq 50$
- The sum of the ℓs across all test cases is at most 200.
- Each character in the expression is a space, a lowercase English letter, a digit, a parenthesis, a bracket, or a brace.
- Each left parenthesis, bracket, or brace has a corresponding right parenthesis, bracket, or brace.

Scoring

Note: New tests may be added and all submissions may be rejudged at a later time. (All future tests will satisfy the constraints.)

- You get 90 ❤️ points if you solve all test cases where:
  - There are no nested parentheses, brackets, or braces.
- You get 80 🔴 points if you solve all test cases.

Clarifications

Report an issue

No clarifications have been made at this time.