Micro1 Programming - Dec 2024

1. Coding exercise

You are developing a text editor's undo feature. The editor keeps track of text modifications where each operation is either an insert or delete of a character. Implement a system that processes these operations and supports an undo function. The system should maintain a history of operations and be able to revert the last operation when requested. For each operation, you need to either: 1)
Insert a character at the end of the text, or 2) Delete the last character if it exists, or 3) Undo the last operation if there is any operation to undo. After each operation, return the current text content.

EXAMPLE 1
Input: l'insert(h)', 'insert(i)', 'delete()',
'undo () ']
Output: ['h', 'hi', 'h', 'hi']
Explanation:First two operations insert 'h' and 'i', delete removes 'i', undo restores the deleted 'i'

EXAMPLE 2
Input: l' insert(a)', 'insert(b)', 'undo()',
'undo()', 'undo()']
Output: l'a', 'ab', 'a', "', '']
Explanation:Inserts 'a' and 'b', first undo removes 'b', second undo removes 'a', third undo has no effect as history is empty

Requirements
1 Implement a solution that processes three types of operations:
insert(char), delete), and undo)
2. Each insert operation adds a character at the end of the text
3. Each delete operation removes the last character if text is not empty
4.Each undo operation reverts the most recent insert or delete operation
5.Return the current text content after each operation
6.Handle edge cases like undo with no previous operations
7.Handle multiple consecutive undo operations correctly
8.Maintain operation history efficiently

2. Problem Statement: Largest Piece of a Rectangular Board

A machine cuts a rectangular board into pieces based on vertical and horizontal cuts. The machine receives the following inputs:

1. `s`: A string representing the size of the board in the form `"width,height"`.
2. `x`: An array of coordinates where vertical cuts are made.
3. `y`: An array of coordinates where horizontal cuts are made.

Write a function `getLargestPiece(s, x, y)` that calculates and returns the **area of the largest piece** of the board after all the cuts are made.

Assumptions:

- The width of the cuts is negligible.
- The board is divided based on the coordinates provided in `x` (vertical cuts) and `y` (horizontal cuts).

Examples:

1. `getLargestPiece("9,9", [3,6], [3,6])` → 9
2. `getLargestPiece("10,10", [8,2], [2,8])` → 36
3. `getLargestPiece("10,10", [3], [])` → 70
4. `getLargestPiece("10,10", [], [])` → 100
5. `getLargestPiece("100,70", [25], [])` → 5250
6. `getLargestPiece("100,70", [], [35])` → 3500
7. `getLargestPiece("100,70", [50], [])` → 3500
8. `getLargestPiece("100,70", [80,20], [35])` → 2100

3. Problem Statement: Vending Machine Change

A vending machine has the following denominations: 1¢, 5¢, 10¢, 25¢, 50¢, and $1. Your task is to write a program that returns change using the least number of coins or notes.

Function Description:
Write a function `getChange(M, P)` where:

- `M` is the amount of money inserted into the machine.
- `P` is the price of the item selected.

The function should return an array of integers representing the number of each denomination to return in the order `[1¢, 5¢, 10¢, 25¢, 50¢, $1]`.

Examples:

1. `getChange(5, 0.99)` → `[1, 0, 0, 0, 0, 4]`
2. `getChange(3.14, 1.99)` → `[0, 1, 1, 0, 0, 1]`
3. `getChange(3, 0.01)` → `[4, 0, 2, 1, 1, 2]`
4. `getChange(4, 3.14)` → `[1, 0, 1, 1, 1, 0]`
5. `getChange(0.45, 0.34)` → `[1, 0, 1, 0, 0, 0]`

Constraints:

- The machine always returns the least number of coins/notes.