Name: ____________________________**W#:** ____________________________**Signature:** ____________________________

CSCI 141 - Final Exam, Fall 2025

Overview

There are five parts to this exam: four functions (10 points each) and a main program (4 points), for a total of 44 points. The specifications are given on this paper document, while only the example inputs are included in the code. I recommend implementing the five parts in the following order:

1. `peter`
2. Main program (at the bottom of `final.py`)
3. `lengthy_words`
4. `that_stacks`
5. `rainier`

Problems

1. `peter(text, chars, count)` (10/44 points)

   Return a string that has the same contents as `text`, except that every instance of each character in `chars` is repeated `count` times. Preconditions: `text` and `chars` are strings; count is an integer ≥ 0.

   Examples:

   ```
   peter("hello world", "o", 3) => 'hellooo wooorld'
   peter("banana boat", "an", 2) => 'baannaannaa boaat'
   peter("Babbage", "ag", 0) => 'Bbbe'
   ```

2. Main program (4/44 points)

   The main program takes 3 or more command line arguments, and prints the result of calling the `peter` function as follows:

   - The first argument is the string of characters to repeat; for the main program, this may not include whitespace.
   - The second is the repetition count.
   - The third and all remaining arguments thereafter are joined with spaces into a single string that is the input text for the `peter` function.

   Be sure to put your main program code inside a main guard.

   Examples:

   ```
   >>> %Run final.py o 3 hello
   hellooo
   >>> %Run final.py o 3 hello world
   hellooo wooorld
   >>>
   >>> %Run final.py wu 4 aw cute cat
   awwww cuuuute cat
   >>>
   ```

3. `lengthy_words(text)` (10/44 points)

   Return a dictionary mapping integer word lengths to the count of words in `text` of that length. For example, `lengthy_words("two banana bot")` yields `{3: 2, 6: 1}`, because it has two three-letter words and one six-letter word. Precondition: text is a string of words separated by spaces.

   More examples:

   ```
   lengthy_words("a bc def") => {1: 1, 2: 1, 3: 1}
   lengthy_words("a b c") => {1: 3}
   lengthy_words("one two three four") => {3: 2, 5: 1, 4: 1}
   lengthy_words("") => {}
   ```

4. `that_stacks(papers)` (10/44 points)

   Given a list of 2-tuples each specifying the length and width of a rectangle, determine whether they can be stacked in the order given without any rectangle extending beyond the edges of the ones below it. In other words, check whether each rectangle can fit inside (perhaps exactly) the previous one. Rectangles may be rotated 90 degrees, which has the effect of swapping their width and length. Precondition: papers is a list of one or more 2-tuples of positive numbers.

   Examples:

```
that_stacks([(4, 4), (3, 3), (2, 2)]) => True
# because each rectangle fits inside the one before it

that_stacks([(4, 2), (3, 2), (1, 2)]) => True

that_stacks([(4, 2), (2, 3), (1, 3)]) => True
# because you can rotate the second and third ones 90 degrees and they fit

that_stacks([(4, 2), (5, 5)]) => False
# because the second rectangle does not fit inside the first

that_stacks([(2, 6), (2, 5), (5, 3)]) => False
# because the third one doesn't fit inside the second
```

5. `rainier(rainfalls)` (10/44 points)

   Given a list of daily rainfall totals, measured in millimeters, find and return the longest continguous stretch of days in which each day had at least as much rain as the prior one. If there are multiple longest sequences, return the first one. Precondition: rainfalls is a nonempty list of numbers. Examples:

```
rainier([5, 4, 2, 3]) => [2, 3]
rainier([1, 2, 2, 1]) => [1, 2, 2]
rainier([1]) => [1]
rainier([1, 2, 1, 3]) => [1, 2] # returns the first length-2 sequence
```