Problem Set: List and String Manipulations with Functions and Loops

Task 1: Working with Lists

Write a program that performs the following:

- 1. Create a list of numbers from 1 to 20.
- 2. Using a loop, remove all numbers divisible by 3 from the list.
- 3. Print the updated list.

Task 2: String Analysis

Create a program to analyze a user-provided string:

- 1. Ask the user to input a sentence.
- 2. Count and print the number of words in the sentence.
- 3. Count and print the number of vowels in the sentence.

 (Hint: Use the string method `.lower()` to handle case insensitivity.)

Task 3: Using List Methods

Write a function `find_second_largest(numbers)` that:

- 1. Takes a list of integers as input.
- 2. Returns the second-largest number in the list.
- 3. Handles cases where the list has duplicates of the largest number.

Task 4: String Methods and Loops

Write a function 'reverse_and_capitalize(sentence)' that:

- 1. Takes a string as input.
- 2. Reverses the order of the words in the string.
- 3. Converts all the letters in each word to uppercase.
- 4. Returns the modified string.

Example:

Input: 'hello world'
Output: 'WORLD HELLO'

Task 5: Cumulative Sum in a List

Write a function `cumulative_sum(numbers)` that:

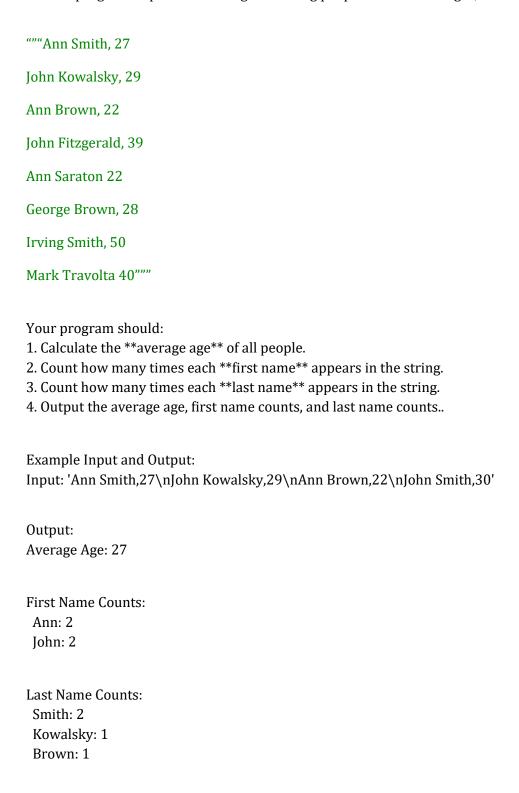
- 1. Takes a list of numbers as input.
- 2. Returns a new list where each element at index `i` is the sum of elements from index `0` to `i` in the input list.

Example:

Input: [1, 2, 3, 4]
Output: [1, 3, 6, 10]

Task 6: Average Age and Name Counts

Write a program to process a string containing people's names and ages, where the string is:



Task 7: Using Loops with String Methods

Write a program that:

- 1. Prompts the user for a list of words (comma-separated).
- 2. For each word:
- Remove any leading or trailing whitespace.
- Check if it starts and ends with the same letter (ignoring case).
- Print the word if the condition is met.

Task 8: Dictionary from List

Write a function `count_occurrences(items)` that:

- 1. Takes a list of strings as input.
- 2. Returns a dictionary where the keys are the unique strings, and the values are the counts of each string in the list.

Example:

```
Input: ['apple', 'banana', 'apple', 'orange', 'banana', 'banana']
Output: {'apple': 2, 'banana': 3, 'orange': 1}
```

Task 9: Sentence Modifier

Write a function `replace_vowels(sentence, char)` that:

- 1. Takes a string 'sentence' and a character 'char' as input.
- 2. Replaces all the vowels in the sentence with 'char'.
- 3. Returns the modified sentence.

Example:

```
Input: ('Hello, world!', '*')
Output: 'H*ll*, w*rld!'
```

Task 10: Sorting and Filtering

Write a function `filter_and_sort(numbers, threshold)` that:

- 1. Takes a list of integers and a threshold as input.
- 2. Filters out all numbers less than the threshold.
- 3. Returns the filtered numbers sorted in descending order.

Example:

```
Input: ([10, 4, 7, 3, 8, 15], 7)
Output: [15, 10, 8, 7]
```