Homework No. 05

Due: 23:59, 26 November, 2023

Max points: 100

Rules

- **No late homeworks.** A penalty of 10 points is applied for each day.
- **No plagiarism.** Collaboration is encouraged, but copying someone else's work without proper attribution is not admitted and invalidates the submission. A penalty is applied to all parties included.

Submission procedure

- Each problem solution should be saved in a separate file. The following naming convention should be used: `problem{number}.{extension}`. For example, `problem1.py` or `problem1.pdf`.
- At the start of each file, homework number, student full name and problem number should be mentioned. For example:

```
"""
Homework 5
Name: John Doe
Problem 1
"""
```

- Solution files should be uploaded to YSU Moodle. Alternatively, you can commit your solutions to a Git repository and provide the repository URL on Moodle.

Problem 1: Frequency Counter [10 points]

Function: `count_frequency` Description: Write a function `count_frequency` that takes a list of strings and returns a dictionary where each key is a unique string from the list and its value is the frequency of its occurrence in the list.

Example

```python
def count_frequency(strings):
    pass

strings = ["apple", "banana", "apple", "orange", "banana", "apple"]
print(count_frequency(strings)) # {"apple": 3, "banana": 2, "orange": 1}
```

Problem 2: Inverse Mapping [20 points]

Function: `inverse_dict` Description: Create a function `inverse_dict` that inverts the keys and values of a given dictionary. The function should handle the scenario where multiple keys have the same value, storing the results as lists of keys under each value.

Example

```python
def inverse_dict(d):
    pass

d = {"a": 1, "b": 2, "c": 1}
print(inverse_dict(d)) # {1: ["a", "c"], 2: ["b"]}
```

Problem 3: Merge and Sum Dictionaries [20 points]

Function: `merge_sum_dicts` Description: Write a function `merge_sum_dicts` that merges two dictionaries. If the same key exists in both dictionaries, sum their values.

Example

```python
def merge_sum_dicts(d1, d2):
    pass

d1 = {"a": 5, "b": 10, "c": 3}
d2 = {"b": 7, "c": 1, "d": 4}
print(merge_sum_dicts(d1, d2)) # {"a": 5, "b": 17, "c": 4, "d": 4}
```

Problem 4: Anagram check [20 points]

Function: `are_anagrams` Description: Write a function `are_anagrams` that takes two strings and returns `True` if they are anagrams and `False` otherwise. Two strings or sequences in general are considered anagrams if we can rearrange the letters / elements of one to obtain the other.

Ignore spaces and letter capitalization.

Example

```python
def are_anagrams(s1, s2):
    pass

print(are_anagrams("listen", "silent")) # True
print(are_anagrams("I am Lord Voldemort", "Tom Marvolo Riddle")) # True
```

Problem 5: Are strings shifted? [30 points]

Function: `are_shifted` Description: Write a function `are_shifted` that takes two strings and returns `True` if the second string is a shifted version of the first string and `False` otherwise. We say that a string is shifted if the characters of second string can be obtained by changing the value of each character in the first string by a fixed number of positions in the alphabet.

For example, `"bcd"` is the shifted version of `"abc"` with shift value of 1. `"Hello"` and `"Lipps"` are shifted versions of each other with shift value of 4.

Ignore the letter capitalization.

Example

```python
def are_shifted(s1, s2):
    pass

print(are_shifted("abc", "bcd")) # True
print(are_shifted("abc", "xyz")) # True
print(are_shifted("Hello", "Lipps")) # True
print(are_shifted("Hello", "World")) # False
```