PIC 16: Homework 2 (due 4/16 at 10pm)

In every question, you are given an empty function with a `docstring` and you have to define it correctly. In every question, you are given simple test cases, but should be careful to think of more complicated ones in case these break your code! How should your answers be submitted?

- Download `hw2.py`. Do NOT change the filename.
- Replace `mjandr` by your name.
- Delete the `pass` statements and type your answers. Do NOT change the function names.
- Do NOT include your test cases.

  This will hinder the grading process and might result in `0` points.

  I suggest solving the homework in one `.py` file, and then carefully transferring your work to `hw2.py`. Confirm you didn't mess up by running your test cases *at the bottom*, and then make sure to *delete them after this*.
- Submit `hw2.py` to CCLE.

Question 1 and 2 will be graded by running fairly similar test cases to the ones provided.
Question 3 will be tested on a far simpler file (similar to `hw2.txt` which is posted on CCLE).

1. First, I need to give a definition which will be used in this question.

   Definition. Suppose `d` is a dictionary whose keys and values are `int`s.

   A *path* in `d` is a list `[(k1,v1), (k2,v2), (k3,v3), ...]` such that:

   - `d[k1] == v1, d[k2] == v2, ...`
   - `v1 == k2, v2 == k3, ...`
   - no two `k`s in the list are the same.

   The *length of the path* is defined to be the length of the list.

   Example. Suppose `d == {1:2, 2:3, 3:4, 4:1}`.

   - `[(1,3), (3,4)]` is *not* a path in `d` because `d[1] != 3`.
   - `[(1,2), (3,4)]` is *not* a path in `d` because `2 != 3`.
   - `[(1,2), (2,3), (3,4), (4,1), (1,2)]` is *not* a path in `d` because `1` is used as a `k` twice.
   - `[(1,2), (2,3)]` is a path in `d` of length `2`.
   - `[(1,2), (2,3), (3,4), (4,1)]` is a path in `d` of length `4`.
   - `[(3,4), (4,1), (1,2)]` is a path in `d` of length `3`.
   - `[]` is a path in `d` of length `0`.

Fill in the definition for the following:

```
def longest_path_length(d):
    """Returns the length of the longest path in d."""

    pass
```

I'm fine with you defining another function before this one (this is what I did).

Test:

```
d = {1:2, 2:3, 4:5, 5:6, 6:7, 8:9}
print(longest_path_length(d))
```

should print `3`.

Test:

```
d = {1:2, 2:3, 3:1}
print(longest_path_length(d))
```

should print `3`.

Test:

```
d = {1:2, 2:3, 3:4, 5:1, 6:1, 4:2}
print(longest_path_length(d))
```

should print `5`.

Test:

```
d = {}
print(longest_path_length(d))
```

should print `0`.

2.
```
def large_value_keys(d, N):
    """Returns a list containing various keys in d.

    d is a dictionary who values are ints. N is an int.
    The list contains keys k whose corresponding value d[k] is bigger than N.
    The keys are arranged in order of largest value to smallest value.
    """

    pass
```

Test:

```
d = {'one':1, 'two':2, 'three':3, 'four':4,
     'five':5, 'six':6, 'seven':7, 'eight':8}
print(large_value_keys(d, 2))
```

should print `['eight', 'seven', 'six', 'five', 'four', 'three']`.

3. Download the plain text of Agatha Christie's "The Mysterious Affair at Styles" from Project Gutenberg and save it in the same folder as your Python code.

   WARNING: the unicode characters are slightly incorrect on this pdf.
   They are correct in the `.py` file.

```
def count_words(filename):
    """Creates a dictionary from a .txt file counting word occurrences.

    For each word in the text file, there's a key.
    The corresponding value is the number of occurrences of the word.

    https://docs.python.org/3.7/library/stdtypes.html#string-methods

    Capitals are converted to lower case
    so that 'The' does not show up as a key.

    Dashes are replaced with spaces
    so that 'them-at' does not show up as a key.
    Both the short dash (-) and long dash (-) are dealt with.

    Non-alphabetic characters are stripped from words
    so that '"espionage?"' does not show up as a key.
    However, both "paper's" and "papers" can show up as keys.
    """

    with open(filename, encoding='utf-8') as f:
        novel = f.read()

    pass
```

   Part of the purpose of this question is to show you that getting such processing correct is quite annoying/difficult. However, when it comes to the grading, I won't be annoying: look at how much simpler `hw2.txt` is! There's a Python library called `re` to help with such processing. But I'd prefer you to use the string methods linked. Reading such documentation will be a useful learning experience.

   Test:

```
d = count_words('863-0.txt')
print(d['the'])
print(d['i'])
print(len({k for k in d if d[k] == 1}))
print(large_value_keys(d, 600))
```

   should print

```
2843
1704
2665
['the', 'i', 'to', 'of', 'a', 'and', 'it', 'that', 'you', 'was', 'in', 'he']
```