Question 1. [8 MARKS]

Each row in the table below shows a series of statements to be interpreted by the Python shell. For each, describe any requirements that must be met in order for the statements to execute without error. For example, you might say the "x must be a float that is greater than 1." If there is no way for the code to run without error, write "impossible" and say why.

Python statements	**Requirements (or "impossible" plus reason)**
`for (a, b) in m.items():` `    print a % b`	
`d = {1: 5, "abc": "hello", 3: 77}` `d[p] = q`	
`L = [9, 11, 3, 4, 8, 23, -5, 14]` `L[i] = j`	
`T = ("Spring", "almost", "here")` `T[s] = t`	

Solution:

(a) The code requires the following:

- m must be a dictionary
- m's keys and values must be numbers (they needn't be specifically ints)
- no value associated with a key in m can be 0

Many students described the conditions in terms of variables a and b. This is not as good an answer. You could make sure that there is an int a defined, and an int b defined with a non-zero value, then insert the code fragment and have it cause an error.

(c) The code requires the following:

- p must be immutable
- q must be defined (it doesn't matter what type or value it has)

(c) The code requires the following:

- i must be and int
- i must be between 0 and 7 inclusive. (Actually, i can be as low as -8.)
- j must be defined (it doesn't matter what type or value it has)

(d) The code requires the following:

- Impossible
- Because tuples are immutable.

Question 2. [10 MARKS]

Part (a) [6 MARKS] Complete the following function according to its docstring description. Note that the word maternal means "relating to a mother."

```
def maternal_list(mother_of, name):
    Return a list where the first item is str 'name' and the subsequent items are the names
    of maternal ancestors of 'name', in order from youngest to oldest, as found in dict
    'mother_of'. Each key in 'mother_of' is the name of a child and the corresponding value
    is that child's mother.  For example, if the dictionary is
            {"jo": "ani", "jane": "di", "lina": "jo", "ani": "frida", "di": "cate"}
    and name is "lina", then the result should be
            ["lina", "jo", "ani", "frida"]'''
```

Solution:

```
    list = [name]
    while name in mother_of:
        mother = mother_of[name]
        list.append(mother)
        name = mother
    return list
```

Part (b) [4 MARKS] Fill in the table below with four good test cases for `maternal_list`. For each, give the specific values for `mother_of` and `name` that you propose, and the purpose of the test case, *e.g.,* "name with even length" (although that's not a good test case for this function!).

Your cases should be different form each other, and should each test something significant.

`mother_of`	`name`	Purpose of this test case

Solution:

Here are a few good test cases you might have used:

`mother_of`	`name`	Purpose of this test case
`{}`	`"sofia"`	empty dictionary
`{"a": "b"}`	`"sofia"`	name not in dictionary
`{"a": "b", "p": "q"}`	`"p"`	name in dictionary; only one ancestor
`{"a": "b", "b": "c", "c": "d"}`	`"a"`	name in dictionary; several ancestors
`{"a": "b", "p": "q"}`	`"q"`	name in dictionary, but only as a mother, not as a child

Question 3. [7 MARKS]

Complete the following function according to its docstring description.

```
def num_comments(f):
    '''f is an file that has been opened for reading, and it contains python code.  Return
    the number of lines in the file whose first non-whitespace character is "#".'''
```

Solution:

```
    ans = 0
    for line in f:
        line = line.lstrip()
        if len(line) > 0 and line[0] == "#":
            ans = ans + 1
    return ans
```