CSC 108H5 S 2019 MOCK Test 2
Duration — 50 minutes
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Last Name: _____ **First Name:** _____

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above and read the instructions below.)
Good Luck!

This midterm consists of 5 questions on 8 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*

- Comments are not required except where indicated, although they may help us mark your answers.
- No error checking is required: assume all user input and all argument values are valid.
- If you use any space for rough work, indicate clearly what you want marked.
- You may use a pencil; however, work written in pencil will not be considered for remarking.

1: _____/ 6
2: _____/ 4
3: _____/ 6
4: _____/ 4
5: _____/ 5

TOTAL: _____/25

Question 1. [6 marks]

Assume each of the pieces of code below is being entered into the Python console. Each subquestion is independent of the others.

In each space, write what would be displayed on the console, if anything. If a line of code would not cause anything to be displayed, leave the space blank.

If the code would cause an error, write ERROR, and provide a brief EXPLANATION.

Part (a) [1 mark]

```
>>> lst = [1, 2, 3, 4]
>>> lst2 = lst.append(5)
>>> print(lst)


>>> print(lst2)
```

Part (b) [1 mark]

```
>>> lst = ['a', 'b', 'c']
>>> lst2 = ['d']
>>> print(lst + lst2)



>>> x = lst.append('e')
>>> print(x + lst2)
```

Part (c) [2 marks]

```
>>> d = {(1, 2): 'ab', (3, 4): 'cd', (5, 6): 'ef'}
>>> print(d[(1, 2)])



>>> print(d[(1, 2)][0])



>>> d[(13, 24)] = d[(1, 2)] + d[(3, 4)]
>>> print(d)



>>> print(d[0])
```

Part (d) [2 marks]

```
>>> L = [[1, 2], ['a', 'b'], [0, 0, 0]]
>>> print(len(L))


>>> print(L[1][1])


>>> L2 = L
>>> L2.insert(1, ('j', 'k'))
>>> print(L)


>>> print(L2)
```

Question 2. [4 marks]

For the following function, add an appropriate doctest example to complete the docstring. Then, complete the function according to the docstring.

```
def get_A_students(marks: dict) -> list:
    """Given a dictionary where the keys are student IDs and values are integers
    that represent the percentage grade each student got in a course,
    return a list of the IDs of all students who got an A (80 or above) in the course.




    """
```

Question 3. [6 marks]

Consider the following function docstring and sample doctest.

```
def all_fluffy(d: dict) -> bool:
    """Return True iff every letter key in d is fluffy.
    Fluffy letters are those that appear in the word 'fluffy'.
    """
```

Part (a) [2 marks]

Here is an *incorrect* implementation of this function.

```
def all_fluffy_v1(d: dict) -> bool:
    for ch in d:
        if ch in 'fluffy':
            return True
        else:
            return False
```

Fill in the table below for the above function with appropriate test cases to discover which cases the function does not work for, and which it does work for. The parameter value is the value we call our function with to test it. The expected return value is what the function should return for this parameter value, according to the docstring. The actual return value should be the value that the code we provided for you above returns.

Parameter Values (d)	Expected Return Value	Actual Return Value

Part (b) [2 marks]

Here is another implementation of this function.

```
def all_fluffy_v2(d: dict) -> bool:

    for ch in d:
        if ch in 'fluffy':
            result =  True
        else:
            result =  False

    return result
```

Fill in the table below for the above function with appropriate test cases to discover which cases the function does not work for, and which it does work for.

Parameter Values (d)	Expected Return Value	Actual Return Value

Does this code work? If not, explain why not.

Part (c) [2 marks]

Here is another implementation of this function.

```
def all_fluffy_v3(d: dict) -> bool:

    result = True
    for ch in "fluffy":
        if ch not in d:
            result = False

    return result
```

Fill in the table below for the above function with appropriate test cases to discover which cases the function does not work for, and which it does work for.

Parameter Values (d)	Expected Return Value	Actual Return Value

Does this code work? If not, explain why not.

Question 4. [4 marks]

Complete the function below according to the docstring.

```
def total_comments(file_name):
    '''
    :param file_name: String file_name
    :return: the number of lines in file_name containing a # character

    Example: If file called "song.txt" contains the following:

    This is the song that never ends
    # Yes it goes on and on my friends
    Some people started singing it
    # Not know what it was # said Sadia
    And they'll continue singing it forever just because...

    Then total_comments("song.txt") == 2
    '''

    # YOUR CODE GOES HERE
```

Question 5. [5 marks]

```
def total_comments2(file_list):
    '''
    Write a file called file_list+".comments" which contains a report consisting of
    the number of # comments in files listed in file_list.

    :param file_list: A string, the name of a file with one file name per line.
    :return: The total number of comments in all files listed in file_list

    Example:
    Assume total_comments("song.txt") == 2, total_comments("something.txt") == 7
           total_comments("big_file.txt") == 3
    and file "theList" consists of (one file name per line)

    song.txt
    something.txt
    big_file.txt

    Then total_comments2("theList") == 12

    and newly created file "theList.comments" has

    song.txt      2
    something.txt 7
    big_file.txt  3

    Note: the space between filename and number is a tab character "\t"

    '''

    # YOUR CODE GOES HERE
    # HINT: Use total_comments(file_name) from the previous question as a helper function
```

Short Python function/method descriptions:

```
__builtins__:
  sorted(sequence) -> sequence
     Return the elements in the given sequence in sorted (non-decreasing) order.
  range([start], stop, [step]) -> list-like-object of int
    Return the integers starting with start and ending with stop - 1 with step
    specifying the amount to increment (or decrement).  If start is not specified,
    the sequence starts at 0.  If step is not specified, the values are incremented by 1.
  str(x) -> str
    Return an object converted to its string representation, if possible.
str:
  S.find(sub[,i]) -> int
    Return the lowest index in S (starting at S[i], if i is given) where the
    string sub is found or -1 if sub does not occur in S.
  S.split([sep]) -> list of str
    Return a list of the words in S; use string sep as the separator and
    any whitespace string if sep is not specified.
list:
  L.append(object) -> NoneType
    Append object to end of L.
  L.extend(iterable) -> NoneType
     Extend list L by appending elements from the iterable. Strings and lists are
     iterables whose elements are characters and list items respectively.
  L.pop([index]) -> item
    Remove and return item at index (default last).
    Raises IndexError if list is empty or index is out of range.
  L.index(value, [start, [stop]]) -> integer
     Return first index of value. Raises ValueError if the value is not present.
dict:
  D[k] -> object
    Return the value associated with the key k in D.
  D.get(k, [optional=None]) -> object
    Return the value associated with the key k in D.
    If this doesn't exist, return the second argument (by default this is None).
  D.keys() -> list-like-object of object
    Return the keys of D.
  D.values() -> list-like-object of object
    Return the values associated with the keys of D.
  D.items() -> list-like-object of tuple of (object, object)
    Return the (key, value) pairs of D, as 2-tuples.
file:
  F.open(s, [optional='r']) -> file object
      Open and return the file named the given filename s. By defualt,
      file is opened with mode 'r' for reading. If mode 'w' is given, the file is opened for writing.
  F.read() -> str
     Read the contents of the file and return as one string. Each line
     in the file is separated by the newline character \n.
  F.readline() -> str
     Read the next line in the file and return as a string ending
     the newline character \n.
  F.write(s) -> int
     Write the given string s to the file F. Return the number of characters written.
```