CSC 108H1 F 2017 Midterm Test
Duration — 50 minutes
Aids allowed: none

UTORid: _____

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

Lecture Section: **(circle one)**: L0301 (MWF1) L0401 (MWF2)
Instructor: Tom Fairgrieve Tom Fairgrieve

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)
Good Luck!

This midterm is double-sided, and consists of 6 questions and a list of function/method descriptions. *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.
- Do not remove any pages from the exam booklet.
- You may use a pencil; however, work written in pencil will not be considered for remarking.

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

TOTAL: _____/23

Question 1. [6 marks]

Beside each code fragment in the table below, write what is printed when the code fragment is executed. If the code would cause an error, write ERROR and give a brief explanation.

Code	Output or Cause of Error
`term = 'Fall' + 2017` `print(term)`	

`msg = 'ten'`
`value = int(msg)`
`print(value)`	

`for value in range(10, 3, -2):` `  print(value)`	

`L = ['a', 'b']`
`L = L.extend(['c'])`
`print(L)`	

`foods = ['fig', 'egg', 'yam', 'pie']`
`fave = foods[1:][1]`
`print(fave)`	

`result = 'assessment'.count('ss', 2)` `print(result)`	

Question 2. [2 MARKS]

Complete the docstring examples with arguments that will cause the function calls to return the values shown.

```
def midterm_function(s: str, i: int) -> bool:
    """
    Precondition: len(s) >= 1 and  0 <= i < len(s)

    >>> midterm_function(          ,          )
    True
    >>> midterm_function(          ,          )
    False
    """

    return s[i:].isdigit()
```

Question 3. [3 MARKS]

Step 1 of the Function Design Recipe (docstring examples) has been completed for the function `remove_occurrence`. Complete steps 2 and 3 of the Function Design Recipe: Fill in the function header (including the type contract) and write a good description.

Do not write the function body. Do not include preconditions.

```
def remove_occurrence
    """


    >>> remove_occurrence('cats scat', 'cat')
    's scat'
    >>> remove_occurrence('abcd', 'bc')
    'ad'
    >>> remove_occurrence('happy', 'day')
    'happy'
    """

    # DO NOT WRITE THE BODY OF THIS FUNCTION
```

Question 4. [4 marks]

Complete the following function according to its docstring.

```
def cooking_time(weight: float, stuffed: bool) -> int:
    """Return the cooking time (in minutes) for a turkey of a given weight
    (in pounds) that may or may not be stuffed, according to the times in
    the following table:

    weight of turkey                cooking time when not stuffed
    ----------------                -----------------------------
    under 14 pounds                 195 minutes
    14 to 20 pounds, inclusive      240 minutes
    over 20 pounds                  270 minutes

    Add 30 minutes to the cooking time when the turkey has been stuffed.

    Precondition: weight > 0

    >>> cooking_time(18.5, False)
    240
    >>> cooking_time(13.3, True)
    225
    >>> cooking_time(14.0, True)
    270
    """
```

Question 5. [5 marks]

Complete the following function according to its docstring.

```
def upper_lower_difference(s: str) -> int:
    """Return the difference between the number of uppercase and lowercase
    letters in s (the number of uppercase minus the number of lowercase).

    >>> upper_lower_difference('Hello99')
    -3
    >>> upper_lower_difference('LISTEN')
    6
    >>> upper_lower_difference('123HiLo')
    0
    """
```

Question 6. [3 marks]

Fill in the box with the while loop condition required for the function to work as described in its docstring.

```
def find_uppercase_vowel(msg: str) -> int:
    """Return the index of the first uppercase vowel (A, E, I, O, U) in msg,
    or the length of msg if it does not contain any uppercase vowels.

    >>> find_uppercase_vowel('CATS')
    1
    >>> find_uppercase_vowel('PYTHON')
    4
    >>> find_uppercase_vowel('aBCDe')
    5
    """

    i = 0

    while [                                                    ]:
        i = i + 1
    return i
```

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

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

Short Python function/method descriptions:

```
__builtins__:
  int(x: object) -> int
    Convert x to an integer, if possible. A floating point argument will be truncated towards zero.
  len(x: object) -> int
    Return the length of list, tuple, or string x.
  print(values: object) -> None
    Prints the values.
  range([start: int], stop: int, [step: int]) -> 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: object) -> str
    Return an object converted to its string representation, if possible.
str:
  x in s -> bool
    Produce True if and only if string x is in string s.
  S.count(sub: str[, start: int[, end: int]]) -> int
    Return the number of non-overlapping occurrences of substring sub in string S[start:end].
    Optional arguments start and end are interpreted as in slice notation.
  S.find(sub: str[,i: int]) -> 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.isalpha() -> bool
    Return True if and only if all characters in S are alphabetic
    and there is at least one character in S.
  S.isalnum() -> bool
    Return True if and only if all characters in S are alphanumeric
    and there is at least one character is S.
  S.isdigit() -> bool
    Return True if and only if all characters in S are digits
    and there is at least one character in S.
  S.islower() -> bool
    Return True if and only if all cased characters in S are lowercase
    and there is at least one cased character in S.
  S.isupper() -> bool
    Return True if and only if all cased characters in S are uppercase
    and there is at least one cased character in S.
  S.lower() -> str
    Return a copy of the string S converted to lowercase.
  S.replace(old: str, new: str) -> str
    Return a copy of string S with all occurrences of the string old replaced with the string new.
  S.upper() -> str
    Return a copy of the string S converted to uppercase.
list:
  x in L -> bool
    Produce True if and only if object x is in list L
  L.append(item: object) -> None
    Append item to end of list L.
  L.extend(items: iterable) -> None
    Extend list L by appending elements from items. Strings and lists are
    iterables whose elements are characters and list items respectively.
```