CSC 108H1 F 2015 Midterm Test
Duration — 50 minutes
Aids allowed: none

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

Last Name: ______________________ **First Name:** ______________________

Lecture Section: L5101 (W6-9) Instructor: Myrto Papadopoulou

*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 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:	______/ 8
# 2:	______/ 4
# 3:	______/ 4
# 4:	______/ 5
# 5:	______/ 3
TOTAL:	______/24

Question 1. [8 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**
`print(3 + 5 % 2)`	

`num = len([10, 8, 2])`
`print(num)`	

`values = [3, 7, 2]`
`values[-1] = 'random'`
`print(values)`	

`word = 'good luck'`
`word[0] = 'G'`
`print(word)`	

`print(False or ('october' < 'november'))`	

`snowing = False`
`num = 0`
`print(snowing and 5 / num == 0)`	

`word = 'Code!'`
`print(word[5])`	

`a = [13, 4, 7]`
`b = a.append(5)`
`print(a)`	

Question 2. [4 marks]

In the function below, complete (i) the function description in the space provided, and (ii) the example function calls by adding arguments that result in the return values shown. (For each example call, there are several correct answers, and providing any one of them will earn full marks.)

```
def mystery(message):
    """ (str) -> str

    Precondition: len(message) >= 1



    >>> mystery(                                        )
    '2a'
    >>> mystery(                                        )
    'hello'
    """

    if message[0].isdigit():
        return message[::3]
    else:
        return message
```

Question 3. [4 marks]

Read the function body and then complete the docstring. Give a meaningful function name, write the type contract and the description, and give two examples that return different values.

```
def                                          (s):
    """



    """

    i = 0
    while i < len(s):
        char = s[i]
        if not (char.isalpha() or char.isdigit() or char in '@.'):
            return False
        i = i + 1
    return True
```

Question 4. [5 marks]

Complete this function according to its docstring description.

```
def avoid_characters(message, avoid_string):
   """ (str, str) -> str

   Return a string containing all characters in message that are not in the
   avoid_string, in the order they appear in message.

   >>> avoid_characters('Hello world!', 'deer')
   'Hllo wol!'
   >>> avoid_characters('banana', 'dog')
   'banana'
   """
```

Question 5. [3 marks]

Complete this function according to its docstring description.

```
def get_string_info(message, char):
    """ (str, str) -> str

    Precondition: len(char) == 1 and (char in message == True)

    Return the rest of message after the first occurrence of char.

    >>> get_string_info('watermelon', 'r')
    'melon'
    >>> get_string_info('apple', 'p')
    'ple'
    """
```

[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) -> int
    Convert x to an integer, if possible. A floating point argument will be truncated towards zero.
  len(x) -> int
      Return the length of list, tuple, or string x.
  print(value) --> NoneType
    Prints the values.
  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:
  x in s --> bool
    Produce True if and only if x is in s.
  S.count(sub[, start[, end]]) -> 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[,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.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, new) -> 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 x is in list L
  L.append(object) -> NoneType
    Append object to end of list 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.
```