CSC 108H1 S 2019 Midterm Test
Date: Tue 12 Feb 2019
Time: 1:15pm – 2:30pm
Duration: 75 minutes
Aids allowed: none

UTORid: |_|_|_|_|_|_|_|_|

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

Last Name: ____________________ **First Name:** ____________________

Lecture Sections: L0101 and L0102
Instructor: 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 test booklet.

# 1:	______/ 5
# 2:	______/ 4
# 3:	______/ 4
# 4:	______/ 3
# 5:	______/ 5
# 6:	______/ 7
TOTAL:	______/28

Total Pages = 8

Question 1. [5 marks]

Circle the answer that best describes the output from running the following code:

```
lst1 = ['abc', 'def', 'ghi']
lst2 = lst1[:]
lst1[0] = 5
print(type(lst2[0]))
```

(A) `<class 'list'>`

(B) `<class 'int'>`

(C) `<class 'float'>`

(D) `<class 'str'>`

(E) `<class 'NoneType'>`

(F) Nothing is printed because an error occurs

(G) No error occurs, and something else is printed

Circle the answer that best describes the output from running the following code:

```
values = [4, 3, 2, 1].sort()
print(type(values[0]))
```

(A) `<class 'list'>`

(B) `<class 'int'>`

(C) `<class 'float'>`

(D) `<class 'str'>`

(E) `<class 'NoneType'>`

(F) Nothing is printed because an error occurs

(G) No error occurs, and something else is printed

Circle the answer that best describes the output from running the following code:

```
s = ''
print(len(s) == 0 and s[0] == s[-1])
```

(A) `True`

(B) `False`

(C) an `IndexError` occurs

(D) another error occurs

---

abstemious is a word that has all 5 vowels in order. Circle all of the code snippets that would print `aeiou`.

```
s = 'abstemious'
```

(A) `print(s[0] + s[4] + s[6:9])`

(B) `print(s[::2])`

(C) `print(s[:5:4] + 'iou')`

(D) `print('aeiou'[:2] + s[-4:-1])`

(E) `print('AEIOU'.islower())`

(F)
```
s.remove('m')
print('a' + s[4:-1])
```

(G) None of the above will print `aeiou`

Fill in the box on the right to show the output from executing the following code:

```
s = 0
k = 0
while s < 5:
    s = s + k
    k = k + 1
print('k = ', k)
```

k = □

[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.]

Question 2. [4 marks]

Read the function body for the function `midterm_function` given below and then fill in each box to make the docstring examples correct. The result from each function call should match the return value.

```
def midterm_function(s: str, ch: str) -> int:
    """
    Precondition: len(s) > 0, s.isalpha(), s.islower(),
                  len(ch) == 1, ch.isalpha(), and ch.islower().

    >>> midterm_function('badger', 'a')
    □
    >>> midterm_function('badger', 'z')
    □
    >>> midterm_function('badger', □ )
    3
    >>> midterm_function( □ , □ )
    4
    """

    i = 0
    while i < len(s) and s[i] <= ch:
        i = i + 1
    return i
```

Place rough work below the line.

Question 3. [4 marks]

Consider the problem of writing function `in_order` that has the following docstring description:

```
"""Return True if and only if i, j and k are in order from smallest to largest.
Equal numbers are considered to be in order.
"""
```

Several solution attempts are given below. Some are correct and some are incorrect.

Circle the letter in front of each solution attempt that *correctly* implements the function.

```
(A)  def in_order(i: int, j: int, k: int) -> bool:
         if i > j:
             return False
         elif j > k:
             return False
         else:
             return True
```

```
(B)  def in_order(i: int, j: int, k: int) -> bool:
         if i <= j:
             if j <= k:
                 return True
         else:
             return False
```

```
(C)  def in_order(i: int, j: int, k: int) -> bool:
         if i <= j:
             return True
         elif j <= k:
             return True
         else:
             return False
```

```
(D)  def in_order(i: int, j: int, k: int) -> bool:
         if i <= j or j <= k:
             return True
         else:
             return False
```

Question 4. [3 marks]

Complete the following function according to its docstring.

Note that the post office will accept postal codes that contain space(s), but the Python function `is_postal_code` should **not** consider such codes to be valid.

```
def is_postal_code(s: str) -> bool:
    """Return True if and only if s refers to a valid Canadian postal code.

    Canadian postal codes are 6 characters long and contain alternating
    letters and digits, starting with a letter.  Letters may be uppercase
    or lowercase.

    >>> is_postal_code('m5W1e6')
    True
    >>> is_postal_code('m5W 1e6')
    False
    >>> is_postal_code('r2d24u')
    False
    """
```

Question 5. [5 marks]

An index in a list of `int`s is called a **peak index** if and only if it is the index of an `int` that is strictly greater than the `int`s on either side of it in the list. There may be many peak indexes in a given list.

Note that, by this definition, the first and last indexes cannot be peak indexes.

Fill in the boxes below to correctly complete the body of function `get_peak_indexes`.

```
def get_peak_indexes(lst: List[int]) -> List[int]:
    """Return a list of all peak indexes in lst. A peak index is the index of
    any int that is stricly greater than the ints on either side of it in the
    list. The ints at the ends of lst cannot be peaks because the do not have
    ints on both sides.

    >>> test_list = [5, 4, 6, -1, 7, 6]
    >>> get_peak_indexes(test_list)
    [2, 4]
    >>> test_list = [5, 6, 6, 5]
    >>> get_peak_indexes(test_list)
    []
    """

    peaks = 

    for i in range(                                    ):

        if                                            :
            # Append the current index to the peaks list.


    return peaks
```

Question 6. [7 marks]

Follow all steps of the Function Design Recipe to write a function named `replace_multiples_with_zero` that takes a list of integers as the first parameter and an integer as the second parameter. The function should assume that the second parameter is positive. The function should replace each number in the original list that is an exact multiple of the second parameter with the number 0, and leave other numbers unchanged.

Remember that you can check whether one number is a multiple of another number by using the `%` operator. For example, `8 % 4` is `0`, which allows us to conclude that `8` is a multiple of `4`. Also, `8` is not a multiple of `5` as `8 % 5` is `3`.

Your answer should contain a complete docstring for this function that includes a description and two examples with expected results, a function header with type contracts, and a function body. Remember that a doctest involving mutability has this form:

```
>>> lst = [AN EXAMPLE LIST]
>>> function_call(lst, OTHER PARAMETERS)
>>> lst
[EXPECTED MUTATED LIST]
```

[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:

```
  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.
  min(iterable: object) -> object
  min(a, b, c, ...) -> object
      With a single iterable argument, return its smallest item.
      With two or more arguments, return the smallest argument.
  print(values: object) -> None
    Prints the values.
  range([start: int], stop: int, [step: int]) -> list-like-object of int
    Return the integers from start (inclusive) to stop (exclusive) by step increments.  If start
    is not specified, the sequence starts at 0.  If step is not specified, the step is 1.
  str(x: object) -> str
    Return an object converted to its string representation, if possible.
  type(x: object) -> the object's type
    Return the type of the object x.
str:
  x in s -> bool
    Produce True if and only if string x is in string s.
  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.
  L.remove(value: object) -> None
    Remove first occurrence of value from L.  Raises ValueError if the value is not present.
  L.sort() -> None
    Sort the contents of L from smallest to largest.
```