UNIVERSITY OF TORONTO
Faculty of Arts & Science

DECEMBER 2019 EXAMINATIONS

CSC 108 H1F

Duration: 3 hours

Aids Allowed: None

*Do **not** turn this page until you have received the signal to start.* In the meantime, write your name, student number, and UTORid below (please do this now!) and *carefully* read *all* the information on the rest of this page.

First (Given) Name(s):

Last (Family) Name(s):

10-Digit Student Number:

UTORid (e.g., `pitfra12`):

- This final examination consists of 10 questions on 22 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the examination is complete.*
- Answer each question directly on the examination paper, in the space provided, and use a "blank" page for rough work. If you need more space for one of your solutions, use one of the "blank" pages and *indicate clearly the part of your work that should be marked.*
- Comments and docstrings are not required except where indicated, although they may help us mark your answers.
- You do not need to put `import` statements in your answers.
- No error checking is required: assume all function arguments have the correct type and meet any preconditions.
- *Remember that, in order to pass the course, you must achieve a grade of at least **40%** on this final examination.*
- As a student, you help create a fair and inclusive writing environment. If you possess an unauthorized aid during an exam, you may be charged with an academic offence.

MARKING GUIDE

Nº 1:	_____/ 4
Nº 2:	_____/ 4
Nº 3:	_____/ 6
Nº 4:	_____/ 6
Nº 5:	_____/ 7
Nº 6:	_____/ 4
Nº 7:	_____/ 6
Nº 8:	_____/ 7
Nº 9:	_____/11
Nº 10:	_____/20
TOTAL:	_____/75

STUDENTS MUST HAND IN ALL EXAMINATION MATERIALS AT THE END

December 2019 Examinations

CSC 108 H1F

Duration: **3 hours**

Question 1. [4 marks]

Part (a) [1 mark] Complete the function `is_eligible_for_award` according to its docstring below.

```
def is_eligible_for_award(gpa: float, major: str, req_gpa: float, req_major: str) -> bool:
    """Return True if and only if gpa is greater than or equal to req_gpa and
    major is equal to req_major.

    >>> is_eligible_for_award(4.0, 'CS', 3.5, 'CS')
    True
    >>> is_eligible_for_award(4.0, 'Econ', 3.5, 'CS')
    False
    >>> is_eligible_for_award(3.3, 'CS', 3.5, 'CS')
    False
    """
    return
```

Part (b) [3 marks] Complete the function `get_award_winners` according to its docstring below. Your code must call the function `is_eligible_for_award` from Part (A).

```
def get_award_winners(applicants: List[Tuple[float, str, str]], req_gpa: float,
                      req_major: str) -> List[Tuple[float, str]]:
    """Return a new list of students from applicants who are eligible for an
    award, based on req_gpa and req_major. Each student in applicants is of
    the form (gpa, name, major).

    In the new list, award winning students should be of the form (gpa, name)
    and should appear in the same order as they appear in the original list.

    >>> get_award_winners([(3.6, 'JC', 'CS'), (4.0, 'MB', 'CS'), (3.7, 'JW', 'Econ')], 3.2, 'CS')
    [(3.6, 'JC'), (4.0, 'MB')]
    >>> get_award_winners([(3.6, 'JC', 'Bio'), (4.0, 'MB', 'Bio'),
                           (3.7, 'JW', 'Econ')], 3.7, 'Bio')
    [(4.0, 'MB')]
    """
```

December 2019 Examinations

CSC 108 H1F

Duration: **3 hours**

Question 2. [4 marks]

Function `mask_string` has been correctly implemented. Fill in the missing parts of the docstring description and examples so that they match the function's header and body.

```
def mask_string(s: str, mask: List[bool], ch: str) -> str:
    """Return a new string that is the same as [          ]
    but with each character that corresponds to a True element in [          ]
    replaced with [          ].

    Precondition : len(ch) == 1 and [                    ]

    >>> mask_string('mask',[True, True, True, True],'?')
    [          ]
    >>> mask_string('ComSc', [ [  ], [  ], [  ], [  ], [  ], [  ])
    'C--S-'
    """

    masked_string = ''
    for i in range(len(s)):
        if mask[i]:
            masked_string = masked_string + ch
        else:
            masked_string = masked_string + s[i]
    return masked_string
```

Question 3. [6 marks]

In Assignment 1, we split a message into substrings (called tweets), where each tweet had a length equal to MAX_LENGTH characters, except the last tweet which had a length of at least 1 up to MAX_LENGTH (inclusive). For example, for a MAX_LENGTH of 10 and the message `'This is such an amazing note!'`, in A1 we split the message into 3 tweets: `'This is su'`, `'ch an amaz'`, and `'ing note!'`.

You must implement an improved version in which the message is split so that no word is divided across multiple tweets. In this improved version, the message above is split into the 4 tweets with lengths as close to MAX_LENGTH as possible (but not longer than MAX_LENGTH) and spaces omitted from the beginning and end of tweets:
`'This is'`, `'such an'`, `'amazing'`, `'note!'`.

Complete the function `split_message` according to the description above and docstring below. Do not add any code outside of the boxes. Your code must use the provided constant. You may assume that there is exactly one space between words in the message and that there is no whitespace at the beginning or end of the message.

```
MAX_LENGTH = 10

def split_message(msg: str) -> List[str]:
    """Return a new list containing msg split into tweets so that no words are
    divided across multiple tweets. A word is a sequence of non-whitespace
    characters. In the tweets, each pair of words must have a space between them.
    There should not be any spaces at the beginning or end of tweets.

    Precondition: no word in msg is longer than MAX_LENGTH

    >>> split_message('One hundred and one')
    ['One', 'hundred', 'and one']
    """

    # split message into words
    tweets = [                                    ]

    i = 0
    while i < [                                    ]:
        # combine tweet i and tweet i+1 into a potential tweet
        potl_tweet = [                                                    ]

        if [                                                    ]:
            # update tweet i to be potential tweet; remove tweet i + 1
            [                                    ]
        else:
            # couldn't combine tweets, so increment i (tweet i is finalized)
            i = i + 1
    return tweets
```

Question 4. [6 marks]

This problem involves using a point system to score words. Each letter has a point value and is represented in a "letter to points" dictionary, where each key is a lowercase letter and each value is the points corresponding to that letter. An example of a "letter to points" dictionary is:

```
{'a': 1, 'b': 2, 'c': 3, 'd': 2, 'e': 1, 'f': 4, ..., 'z': 10}
```

Words are scored by adding up the points associated with each of their letters. According to the dictionary above, the word `'Bee'` is worth 4 points (2 + 1 + 1) and the word `'bEAd'` is worth 6 points (2 + 1 + 1 + 2). Only lowercase letters appear in "letter to points", so uppercase letters are scored based on the corresponding lowercase letter's points.

Consider this example file with one word per line:

```
a
BE
bee
Bee
bEAd
```

Each line in the file contains a unique word composed only of letters. Each word has at least one letter. There are no blank lines in the file. Words are case-sensitive (e.g., `'bee'` and `'Bee'` are considered to be different words.)

Given the example file above opened for reading and the "letter to points" dictionary above, function `build_word_to_score` should return this "word to score" dictionary:

```
{'a': 1, 'BE': 3, 'bee': 4, 'Bee': 4, 'bEAd': 6}
```

Complete the function `build_word_to_score` according to the example above and the docstring below. You may assume the given file has the correct format and the "letter to points" dictionary contains all 26 lowercase letters.

```
def build_word_to_score(word_file: TextIO, letter_to_points: Dict[str, int]) -> Dict[str, int]:
    """Return a "word to score" dictionary, where each key is a word from word_file and
    each value is the score for that word based on the points in letter_to_points.
    """
```

Question 5. [7 marks]

In this question, you will implement two different algorithms for solving the same problem. Given a list of integers in which each item occurs either once or twice, count the number of items that occur twice.

Part (a) [4 marks]

Complete this function according to its docstring by implementing the algorithm below:

1. Use an integer accumulator to keep track of the number of matches.
2. For each item in `lst`, compare it to all other items in `lst` and if the item matches an item other than itself, add to the number of matches.
3. Return the number of duplicates.

Note: you must not use any `list` methods, but you may use the built-in function `len()`.

To earn marks for this question, you must follow the algorithm described above.

```
def count_duplicates_v1(lst: List[int]) -> int:
    """Return the number of duplicates in lst.

    Precondition: each item in lst occurs either once or twice.

    >>> count_duplicates_v1([1, 2, 3, 4])
    0
    >>> count_duplicates_v1([2, 4, 3, 3, 1, 4])
    2
    """
```

Part (b) [3 marks]

Complete this function according to its docstring by implementing the algorithm below:

1. Use a dictionary accumulator to track items and the number of times they occur in `lst`.
2. For each item in `lst`, if it is not already a key in the dictionary, add it along with a value of 1. If it is already a key in the dictionary, increase the value associated with it by 1.
3. Return the difference between the number of items in `lst` and the number of items in the dictionary.

Note: you must not use any `list` methods, but you may use the built-in function `len()`.

To earn marks for this question, you must follow the algorithm described above.

```
def count_duplicates_v2(lst: List[int]) -> int:
    """Return the number of duplicates in lst.

    Precondition: each item in lst occurs either once or twice.

    >>> count_duplicates_v2([1, 2, 3, 4])
    0
    >>> count_duplicates_v2([2, 4, 3, 3, 1, 4])
    2
    """
```

Question 6. [4 marks]

Part (a) [3 marks] Consider using a `List[List[int]]` to represent a square grid of integers (as we did in A2 to represent an elevation map). Complete the function `swap_rows_and_columns` according to its docstring below.

```
def swap_rows_and_columns(grid: List[List[int]]) -> None:
    """Modify grid so that each row in grid is swapped with the corresponding column.
    For example, row 0 is swapped with column 0, row 1 is swapped with column 1, and so on.

    Precondition: each list in grid has the same length as grid and len(grid) >= 2

    >>> G2 = [[1, 2],
    ...       [3, 4]]
    >>> swap_rows_and_columns(G2)
    >>> G2 == [[1, 3],
    ...        [2, 4]]
    True
    >>> G3 = [[1, 2, 3],
    ...       [4, 5, 6],
    ...       [7, 8, 9]]
    >>> swap_rows_and_columns(G3)
    >>> G3 == [[1, 4, 7],
    ...        [2, 5, 8],
    ...        [3, 6, 9]]
    True
    """
```

Part (b) [1 mark] Circle the term below that best describes the running time of the `swap_rows_and_columns` function.

constant	linear	quadratic	something else

Question 7. [6 marks]

For this question, a "person to friends" dictionary (`Dict[str, List[str]]`) has a person's name as a key and a list of that person's friends as the corresponding value. You can assume that each person has at least one friend.

Complete the function `complete_person_to_friends` according to its docstring below. Do not modify the provided code and only write your code after the provided code.

```
def complete_person_to_friends(p2f: Dict[str, List[str]]) -> None:
    """Modify the "person to friends" dictionary p2f so that all friendships are
    bidirectional. That is, if person B is in person A's list of friends, then
    person A must be in person B's list of friends. The friend lists must be
    sorted in alphabetical order.

    >>> friend_map = {'JC': ['MB']}
    >>> complete_person_to_friends(friend_map)
    >>> friend_map == {'JC': ['MB'], 'MB': ['JC']}
    True
    >>> friend_map = {'B': ['A', 'D'], 'A': ['B', 'C', 'D']}
    >>> complete_person_to_friends(friend_map)
    >>> friend_map == {'B': ['A', 'D'], 'A': ['B', 'C', 'D'], \
                       'D': ['A', 'B'], 'C': ['A']}
    True
    >>> friend_map = {'JC': ['MB'], 'MB': ['JW']}
    >>> friend_map == {'JC:': ['MB'], 'MB': ['JC', 'JW'], 'JW': ['MB']}
    >>> complete_person_to_friends(friend_map)
    True
    """
```

```
keys = list(p2f.keys())
for person in keys:
```

Question 8. [7 marks]

Consider the function below:

```
def get_first_even(items: List[int]) -> int:
    """Return the first even number from items. Return -1 if items contains
    no even numbers.  The number 0 is considered to be even.
    """
```

Part (a) [5 marks] Add five more distinct test cases to the table below. Do **not** test if the input, items, is mutated. No marks will be given for duplicated test cases. The six test cases below are not intended to be a complete test suite.

Test Case Description	`items`	*Expected Return Value*
The empty list	`[]`	`-1`

Part (b) [2 marks] A buggy implementation of the `get_first_even` function is shown below. In the table below, complete the two test cases by filling in the three elements of `items`. The first test case should not indicate a bug in the function (the test case would pass), while the second test case should indicate a bug (the test case would fail). Both test cases should pass on a correct implementation of the function. If appropriate, you may use test(s) from Part (a).

```
def get_first_even(items: List[int]) -> int:
    even_number = -1

    for item in items:
        if item % 2 == 0:
            even_number = item

    return even_number
```

	`items`	*Expected Return Value*	*Actual Return Value*
Does not indicate bug (pass)	[, ,]		
Indicates bug (fail)	[, ,]		

Question 9. [11 marks]

In this question, you will write method bodies in classes to represent hotel and hotel room data. Here is the header and docstring for class Room.

```
class Room:
    """A hotel room."""
```

Part (a) [1 mark] Complete the body of this class Room method according to its docstring.

```
    def __init__(self, num: int, max_occupancy: int) -> None:
        """Initialize a room with room number num and a maximum occupancy of
        max_occupancy that is not currently booked.

        >>> r = Room(404, 2)
        >>> r.room_number
        404
        >>> r.max_occupancy
        2
        >>> r.is_booked
        False
        """
```

Part (b) [2 marks] Complete the body of this class Room method according to its docstring.

```
    def book_room(self) -> bool:
        """If this room is not currently booked, update this room's booking
        status to booked and return True. Otherwise, return False and do not
        change the booking status.

        >>> r = Room(401, 4)
        >>> r.book_room()
        True
        >>> r.is_booked
        True
        >>> r.book_room()
        False
        >>> r.is_booked
        True
        """
```

Part (c) [2 marks] Complete the body of this class Room method according to its docstring.

```
    def __str__(self) -> str:
        """Return the string representation of this room.

        >>> r = Room(48, 3)
        >>> str(r)
        'Room 48 has a maximum occupancy of 3 and is not booked.'
        >>> r.book_room()
        True
        >>> str(r)
        'Room 48 has a maximum occupancy of 3 and is booked.'
        """
```

Now consider the class Hotel. Here is the header and docstring for class Hotel. You may assume classes Room and Hotel are in the same file (they do not need to be imported).

```
class Hotel:
    """A hotel that contains rooms."""
```

Part (d) [1 mark] Complete the body of this class Hotel method according to its docstring.

```
    def __init__(self, hotel_name: str) -> None:
        """Initialize a hotel with a hotel_name and an empty rooms list.

        >>> h = Hotel("Sleep EZ")
        >>> h.hotel_name
        'Sleep EZ'
        >>> h.rooms
        []
        """
```

Part (e) [1 mark] Complete the body of this class `Hotel` method according to its docstring.

```
def add_room(self, room: 'Room') -> None:
    """Add this room to the end of this hotel's rooms list.
    You may assume that this room is not already in this hotel's rooms list.

    >>> h = Hotel("Sleep EZ")
    >>> h.rooms
    []
    >>> r = Room(99, 4)
    >>> h.add_room(r)
    >>> h.rooms[0] == r
    True
    """
```

Part (f) [2 marks] Complete the body of this class `Hotel` method according to its docstring.

```
def get_max_occupancy(self) -> int:
    """Return the total occupancy of this hotel.

    >>> h = Hotel("Nighty Night")
    >>> h.add_room(Room(100, 3))
    >>> h.add_room(Room(101, 4))
    >>> h.get_max_occupancy()
    7
    """
```

Part (g) [2 marks] Complete the body of this class `Hotel` method according to its docstring. For full marks on this question, your code should use existing methods (either directly or indirectly) where possible.

```
def __str__(self) -> str:
    """Return the string representation of this hotel.

    >>> h = Hotel("Nighty Night")
    >>> print(h)
    Hotel: Nighty Night
    Max Occupancy: 0
    Rooms:
    >>> r1 = Room(100, 3)
    >>> r2 = Room(101, 4)
    >>> r2.book_room()
    True
    >>> h.add_room(r1)
    >>> h.add_room(r2)
    >>> print(h)
    Hotel: Nighty Night
    Max Occupancy: 7
    Rooms:
    Room 100 has a maximum occupancy of 3 and is not booked.
    Room 101 has a maximum occupancy of 4 and is booked.
    """

    rslt = 'Hotel: {0}\nMax Occupancy: {1}\n'.format(                    ,
                                                                         )
    rslt = rslt + 'Rooms:'
    for room in                    :
        rslt = rslt +
    return rslt
```

December 2019 Examinations

CSC 108 H1F

Duration: **3 hours**

Question 10. [20 marks]

DO NOT put your answers here. You must fill in your answers on the last page of this booklet.

For Questions 1-4, consider the following assignment statement.

```
placements = {
    'NB': [1, 5, 6],
    'Nike': [2, 4],
    'Adidas': [3, 7, 8],
    'Puma': [9, 10]
}
```

1. What does the expression `len(placements) == 10` evaluate to?
   (A) `True`
   (B) `False`
   (C) Nothing, because an error will occur

2. What will this code print?

```
for x in placements:
    for y in x:
        print(y, end=',')
```

   (A) `1,5,6,2,4,3,7,8,9,10,`
   (B) `0,1,2,3,4,5,6,7,8,9,`
   (C) `N,B,N,i,k,e,A,d,i,d,a,s,P,u,m,a,`
   (D) `NB,Nike,Adidas,Puma,`
   (E) Nothing, because an error will occur

3. What will this code print?

```
for x in placements:
    print(type(placements[x][0]))
```

   (A) `<class 'int'>` is printed four times
   (B) `<class 'str'>` is printed four times
   (C) `<class 'list'>` is printed four times
   (D) `<class 'int'>` is printed ten times
   (E) Nothing, because an error will occur

4. What will this code print?

```
del placements['NB']
del placements['Adidas']
print({'Puma': [9, 10], 'Nike': [2, 4]} == placements or 'NB' in placements)
```

   (A) `True`
   (B) `False`
   (C) Nothing, because an error will occur

DO NOT put your answers here. You must fill in your answers on the last page of this booklet.

For Questions 5-7, consider the plain text file `my_file.txt` (below left) and Python code (below right). Assume that the Python code is executed and runs without any errors.

```
Roses are red,
Violets are blue.
I love CSC108!
How about you?
```

```
mario = open('my_file.txt', 'r')
jennifer = mario.readline().strip()
joseph = mario.readlines()[0]
jonathan = mario.read()
```

5. What does the variable `jennifer` refer to?
   (A) The empty string: `''`
   (B) The string: `'Roses are red,\n'`
   (C) The string: `'Roses are red,'`
   (D) The string: `'Violets are blue.'`

6. What does the variable `joseph` refer to?
   (A) The string: `'Roses are red,\n'`
   (B) The string: `'Violets are blue.\n'`
   (C) The string: `'How about you?\n'`
   (D) The list: `['Violets are blue.\n', 'I love CSC108!\n', 'How about you?\n']`
   (E) The list: `['Roses are red,\n', 'Violets are blue.\n', 'I love CSC108!\n', 'How about you?\n']`

7. What does the variable `jonathan` refer to?
   (A) An empty string: `''`
   (B) The string: `'\n'`
   (C) The string: `'How about you?\n'`
   (D) An empty list: `[]`

DO NOT put your answers here. You must fill in your answers on the last page of this booklet.

8. What will this code print?

```
L = [1, 2, 3]
L.reverse()
list.insert(L, 0, 4)
print(L)
```

(A) The list: `[1, 2, 3, 4]`
(B) The list: `[4, 3, 2, 1]`
(C) The list: `[3, 2, 1, 4]`
(D) The list: `[4, 1, 2, 3]`
(E) Nothing, because an error will occur

9. What will this code print?

```
context_to_next = {}
context = ('It', 'was', 'the')
context_to_next[context] = ['best']
context = context[1:] + ('best',)
print(context)
```

(A) `('It', 'was', 'the', 'best')`
(B) `('was', 'the', 'best')`
(C) `('the', 'best')`
(D) Nothing, because an error will occur

10. Variable `greeting` refers to `'hello'`. When the code below runs, which `str` method is called?

```
print(greeting)
```

(A) `__init__`
(B) `__str__`
(C) `__eq__`
(D) `print`

DO NOT put your answers here. You must fill in your answers on the last page of this booklet.

For Questions 11–14, consider this function:

```
def get_first_positive(lst: List[int]) -> int:
    """Return the first positive number in lst, or -1 if there are no positive
    numbers in lst.
    """

    for item in lst:
       if item > 0:
           return item
    return -1
```

11. Which input below results in the *best case* running time of the `get_first_positive` function?

    (A) `lst` refers to the list: `[-1, -8, -9, 5]`
    (B) `lst` refers to the list: `[-2, -8, -9]`
    (C) `lst` refers to the list: `[-6, 3, -9, -10]`
    (D) `lst` refers to the list: `[5, -3, -9, 7, -10]`

12. Which term best describes the *best case* running time of the `get_first_positive` function?

    (A) constant
    (B) linear
    (C) quadratic
    (D) something else

13. Which input below results in the *worst case* running time of the `get_first_positive` function?

    (A) `lst` refers to the list: `[-1, -8, -9, -5]`
    (B) `lst` refers to the list: `[-2, -8, -9, 10]`
    (C) `lst` refers to the list: `[-6, 3, -9, -10]`
    (D) `lst` refers to the list: `[5, -3, -9, -10]`

14. Which term best describes the *worst case* running time of the `get_first_positive` function?

    (A) constant
    (B) linear
    (C) quadratic
    (D) something else

December 2019 Examinations

CSC 108 H1F

Duration: **3 hours**

DO NOT put your answers here. You must fill in your answers on the last page of this booklet.

For Questions 15-20, lists are to be sorted into ascending (increasing) order.

For Questions 15 and 16, consider this list: `[4, 1, 6, 9, 2, 5, 8]`
This is the list after the first and second passes of a sorting algorithm:

```
[4, 1, 6, 9, 2, 5, 8] # after one pass
[1, 4, 6, 9, 2, 5, 8] # after two passes
```

15. Which sorting algorithm is being executed?
    (A) bubble sort
    (B) selection sort
    (C) insertion sort
16. What will the list look like after the third pass?
    (A) `[1, 2, 4, 6, 9, 5, 8]`
    (B) `[1, 4, 6, 9, 2, 5, 8]`
    (C) `[1, 4, 6, 2, 5, 8, 9]`

For Questions 17 and 18, consider this list: `[9, 8, 5, 10, 3, 4, 2, 7]`
This is the list after the first and second passes of a sorting algorithm:

```
[2, 8, 5, 10, 3, 4, 9, 7] # after one pass
[2, 3, 5, 10, 8, 4, 9, 7] # after two passes
```

17. Which sorting algorithm is being executed?
    (A) bubble sort
    (B) selection sort
    (C) insertion sort
18. What will the list look like after the third pass?
    (A) `[2, 3, 5, 10, 8, 4, 9, 7]`
    (B) `[2, 3, 5, 8, 4, 9, 7, 10]`
    (C) `[2, 3, 4, 10, 8, 5, 9, 7]`

For Questions 19 and 20, consider this list: `[6, 3, 0, 8, 1, 2, 7, 5]`
This is the list after the first and second passes of a sorting algorithm:

```
[3, 0, 6, 1, 2, 7, 5, 8] # after one pass
[0, 3, 1, 2, 6, 5, 7, 8] # after two passes
```

19. Which sorting algorithm is being executed?
    (A) bubble sort
    (B) selection sort
    (C) insertion sort
20. What will the list look like after the third pass?
    (A) `[0, 1, 2, 3, 5, 6, 7, 8]`
    (B) `[0, 2, 3, 1, 6, 5, 7, 8]`
    (C) `[0, 1, 3, 2, 6, 5, 7, 8]`

December 2019 Examinations

CSC 108 H1F

Duration: **3 hours**

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

December 2019 Examinations

CSC 108 H1F

Duration: **3 hours**

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Total Marks = 75

CSC108H Fall 2019 Final Exam: Short help descriptions

STUDENTS MUST HAND IN ALL EXAMINATION MATERIALS AT THE END

```
__builtins__:
  abs(x: float) -> float
    Return the absolute value of x.
  float(x: object) -> float
    Convert x to a floating point number, if possible.
  input([prompt: str]) -> str
    Read a string from standard input. The trailing newline is stripped. The prompt string,
    if given, is printed without a trailing newline before reading input.
  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 the list, tuple, dict, or string x.
  max(iterable: object) -> object
  max(a, b, c, ...) -> object
    With a single iterable argument, return its largest item.
    With two or more arguments, return the largest argument.
  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.
  open(name: str[, mode: str]) -> TextIO
    Open a file.  Legal modes are "r" (read) (default), "w" (write), and "a" (append).
  print(value: object, ..., sep=' ', end='\n') -> None
    Prints the values. Optional keyword arguments:
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
  range([start: int], stop: int, [step: int]) -> list-like-object of int
    Return the integers from start (inclusive) to stop (exclusive) 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
    Convert an object into its string representation.
  type(x: object) -> the object's type
    Return the type of the object x.

dict:
  D[k] --> object
    Produce the value associated with the key k in D.
  del D[k]
    Remove D[k] from D.
  k in D --> bool
    Produce True if k is a key in D and False otherwise.
  D.get(k: object) -> object
    Return D[k] if k in D, otherwise return 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[object, object]
    Return the (key, value) pairs of D, as 2-tuples.
```

CSC108H Fall 2019 Final Exam: Short help descriptions

STUDENTS MUST HAND IN ALL EXAMINATION MATERIALS AT THE END

```
file open for reading (TextIO):
  F.close() -> None
    Close the file.
  F.read() -> str
    Read until EOF (End Of File) is reached, and return as a string.
  F.readline() -> str
    Read and return the next line from the file, as a string. Retain any newline.
    Return an empty string at EOF (End Of File).
  F.readlines() -> List[str]
    Return a list of the lines from the file.  Each string retains any newline.

list:
  x in L --> bool
    Produce True if x is in L and False otherwise.
  L.append(x: object) -> None
    Append x to the end of the list L.
  L.extend(iterable: object) -> None
    Extend list L by appending elements from the iterable. Strings and lists are
    iterables whose elements are characters and list items respectively.
  L.index(value: object) -> int
    Return the lowest index of value in L, but raises an exception if value does
    not occur in S.
  L.insert(index: int, x: object) -> None
    Insert x at position index.
  L.pop([index: int]) -> object
    Remove and return item at index (default last).
  L.remove(value: object) -> None
    Remove the first occurrence of value from L.
  L.reverse() -> None
    Reverse the list *IN PLACE*.
  L.sort() -> None
    Sort the list in ascending order *IN PLACE*.
```

CSC108H Fall 2019 Final Exam: Short help descriptions

STUDENTS MUST HAND IN ALL EXAMINATION MATERIALS AT THE END

```
str:
  x in s --> bool
    Produce True if x is in s and False otherwise.
  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.endswith(S2: str) -> bool
    Return True if and only if S ends with S2.
  S.find(sub: str[, start: int[, end: int]]) -> int
    Return the lowest index in S where substring sub is found, such that sub is
    contained within S[start:end], or -1 if sub is not found in S[start:end].
    Default value of start is 0 and end is len(S).
  S.format([args, ...]) -> str
    Return a formatted version of S, using substitutions from args.
    The substitutions are identified by braces ('{' and '}').
  S.index(sub: str[, start: int[, end: int]]) -> int
    Like S.find but raises an exception if sub does not occur in S[start:end].
  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.isdigit() -> bool
    Return True if all characters in S are digits
    and there is at least one character in S, and False otherwise.
  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.join(iterable: object)
    Return the items of iterable concatenated together with S inserted
    in between each pair of items.
  S.lower() -> str
    Return a copy of the string S converted to lowercase.
  S.lstrip([chars: str]) -> str
    Return a copy of the string S with leading whitespace removed.
    If chars is given and not None, remove characters in chars instead.
  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.rstrip([chars: str]) -> str
    Return a copy of the string S with trailing whitespace removed.
    If chars is given and not None, remove characters in chars instead.
  S.split([sep: str]) -> List[str]
    Return a list of the words in S, using string sep as the separator and
    any whitespace string if sep is not specified.
  S.startswith(S2: str) -> bool
    Return True if and only if S starts with S2.
  S.strip([chars: str]) -> str
    Return a copy of S with leading and trailing whitespace removed.
    If chars is given and not None, remove characters in chars instead.
  S.upper() -> str
    Return a copy of the string S converted to uppercase.
```