PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

AUGUST 2016 EXAMINATIONS

CSC 108 H1Y
Instructor(s): Yomna Aly

Duration—3 hours

No Aids Allowed

You must earn at least 30 out of 75 marks (40%) on this final examination in order to pass the course. Otherwise, your final course grade will be no higher than 47%.

Student Number:	
Last (Family) Name(s):	
First (Given) Name(s):	

Do **not** *turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully.*

MARKING GUIDE

# 1:	/10
# 2:	/ 7
# 3:	/ 6
# 4:	/ 5
# 5:	/ 5
# 6:	/ 4
# 7:	/ 4
# 8:	/11
# 9:	/10
# 10:	/ 5
# 11:	/10
TOTAL:	/77

This Final Examination paper consists of 11 questions on 20 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 paper is complete and fill in your name and student number above.*

- 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 user input and all argument values are valid.
- Do not use `break` or `continue` on this exam.
- If you use any space for rough work, indicate clearly what you want marked.
- Do not remove pages or take the exam apart.

Good Luck!

Question 1. [10 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.

See page 3 for Questions

Code	**Output or Cause of Error**
`print(4 * 12 / 2 ** 2 - 2 )`	
`days = [['Mon','Oct',21], ['Wed','Nov',7],` `['Fri','Dec',6]]` `print(days[0][0] == 'Thurs' and days[4] == [])`	
`word = 'Code!'` `print(word[5])`	
`sad = False` `num = 0` `print(7 / num == 1 and not sad)`	
`print(('3' + 3) == 6 and True)`	
`list1 = [1, 2,3]` `list2 = list1.append('10')` `print(list1 == list2 )`	
`cloudy = True` `num = 0` `print(cloudy or 5/num > 3)`	
`b = [1, 3.14]` `b.extend(2.72)` `print(len(b) == 3)`	
`s = 'cat'` `s[0] = 'h'` `print(s)`	

Question 2. [7 MARKS]

Part (a) [4 MARKS]

Complete this function according to its docstring description.

```
def sum_values_above_threshold(value_string, threshold):
    """ (str, int) -> int

    Precondition: value_string.isdigit() returns True

    Return the sum of the individual digits in value_string that are
    greater than threshold.

    >>> sum_values_above_threshold('153382', 4)
    13
    >>> sum_values_above_threshold('12345', 5)
    0
    """
```

Part (b) [3 MARKS]

In the table below, we have outlined one test case for `sum_values_above_threshold`. Add three more test cases that could be part of a complete set of cases for thoroughly testing the function. Do not include duplicate test cases. You must not use the arguments from the docstring examples.

Test Case Description	Arguments	Expected Return Value
only last digit above threshold	'1234', 3	4

Question 3. [6 MARKS]

Each of the following sets of Python statements will result in an error message being displayed when the code is run. Explain briefly the cause of each error in the table below.

Python statements	**Explain briefly why an error message is displayed**

`food_to_count = {'burger': 3, \`
`        'fries': 5, 'salad': 7}`
`food_to_count['salad'] = 1`
`print(food_to_count[2])`	

`foods = ['soup', 'waffle', 'pizza']`
`for food in foods:`
`    food[0] = food[0].upper()`
`print(foods)`	

`grades = [75, 68, 82]`
`grades = grades.extend([90])`
`print(grades[-1])`	

`full_name = ['Jovi', 'Jon', 'Bon']`
`name = full_name[1:] + full_name[0]`
`print(name)`	

`["Temperature", " is "].extend(10)`	

`filename = 'data.txt'`
`file_lines = filename.readlines()`
`print(file_lines[-1])`	

Question 4. [5 MARKS]

Part (a) [5 MARKS] Consider the Python Code below,in the space below write what this code prints when run

```
Part (a): (3 marks)

L = [8, 12, 3]
X = L.sort()
Y = L[:]
L.extend([1])
print(X)
print(id(Y) == id(L))
print(L)

Part (b): (2 marks)

def find_bias(lst):
    """ (list of int) -> int
    """

    bias = 0
    for num in lst:
       if num % 2 == 0:
          return bias + 1
       else:
          return bias - 1
     return bias

my_list = [2, 4, 5, 6]
print('The even/odd bias is:', find_bias(my_list))
```

Question 5. [5 MARKS]

Part (a) [5 MARKS]

Write the body of the function according to its docstring description.

```
def convert_time_to_seconds(time_as_str):
    """ (str) -> int
       Precondition: time_as_str is a str in the format 'h:m:s', with
       0 <= int(h) <= 23 and 0 <= int(m) <= 59 and 0 <= int(s) <= 59

       Return the number of seconds in time_as_str.
       >>> convert_time_to_seconds('1:10:25')
       4225
    """
```

Question 6. [4 MARKS]

Consider the following two function definitions. Beside each code fragment in the table below, write what is printed when the code fragment is executed. Assume all of the code is contained and run in the same Python module.

```
def first_function(values):
    """ (list of int) -> NoneType
    """

    for i in range(len(values)):
        if values[i] % 2 == 1:
            values[i] = values[i] + 1

def second_function(value):
    """ (int) -> int
    """

    if value % 2 == 1:
        value = value + 1
    return value
```

Code	Output

`a = [1, 2, 3]`
`b = 1`
`first_function(a)`
`second_function(b)`
`print(a)`
`print(b)`	

`a = [1, 2, 3]`
`b = 1`
`print(first_function(a))`
`print(second_function(b))`	

Question 7. [4 MARKS]

Part (a) [1 MARK]

The list below is shown after each pass of a sorting algorithm.

```
[7, 1, 4, 5, 2, 6, 3] # intial list
[1, 4, 5, 2, 6, 3, 7] # after one pass
[1, 4, 2, 5, 3, 6, 7] # after two
[1, 2, 4, 3, 5, 6, 7] # after three
[1, 2, 3, 4, 5, 6, 7] # after four
```

Which sorting algorithm is being executed? (circle one) (a) bubble sort (b) selection sort (c) insertion sort

Part (b) [1 MARK]

The list below is shown after each pass of a sorting algorithm.

```
[7, 1, 4, 5, 2, 6, 3] # initial list
[1, 7, 4, 5, 2, 6, 3] # after one pass
[1, 2, 4, 5, 7, 6, 3] # after two
[1, 2, 3, 5, 7, 6, 4] # after three
[1, 2, 3, 4, 7, 6, 5] # after four
[1, 2, 3, 4, 5, 6, 7] # after five
```

Which sorting algorithm is being executed? (circle one) (a) bubble sort (b) selection sort (c) insertion sort

Part (c) [2 MARKS]

List [6, 5, 2, 3, 7, 1, 4] is being sorted using insertion sort. Fill in the blanks to show the list after the next two passes.

After one pass: [6, 5, 2, 3, 7, 1, 4]

After two passes: [5, 6, 2, 3, 7, 1, 4]

After three passes:

After four passes:

Question 8. [11 MARKS]

This question has you write the bodies of two functions. Complete each function according to its docstring.

Part (a) [6 MARKS]

Note: you will most likely not need all of the space on this page.

```
def count_letter_case(L):
    """ (list of list of str) -> list of tuple of int
        Precondition: each str in L is non-empty and contains only alphabetic characters

        Count the number of words in each sublist of L that start with lowercase and uppercase let
        Return a new list where each element is a two-item tuple in which the first item is the nu

        >>> count_letter_case([['apple', 'Banana'], ['PEAR'], [], ['PEACH', 'apRICot', 'plum']])
         [(1, 1), (0, 1), (0, 0), (2, 1)]
    """
```

Part (b) [5 MARKS]

Complete the function body according to its docstring description. You will be marked only on the parts that need to be different from (a).

```
def count_letter_case_mutate(L):
    """ (list of list of str) -> NoneType
        Precondition: each str in L is non-empty and contains only alphabetic characters

        Replace each item in L with a two-item tuple in which the first item is the number of w

>>> data = [['apple', 'Banana'], ['PeAr'], [], ['PEACH', 'apRICot', 'plum']]
>>> count_letter_case_mutate(data)
>>> data
[(1, 1), (0, 1), (0, 0), (2, 1)]
    """
```

Question 9. [10 MARKS]

```
def return_mix(num, word):
    """(int, str) -> object
       Precondition: word contains at least 3 characters.
    """

    if word[2] == 'r':
       return 2 * num
    elif num > 10:
       return (word + word)
    elif word[1:3] == "ab":
       return (2 == 3)
```

Write the return value and type for each of the following function calls:

Function Call	Return Value and Type
`return_mix(5, "Canada")`	
`return_mix(12, "Toronto")`	
`return_mix(16, "Canada")`	
`return_mix(-2, "cabinet")`	
`return_mix(11, return_mix(11, "Can"))`	

Question 10. [5 MARKS]

The code below partially implements a bidirectional version of selection sort (sometimes called 'cocktail sort') where the sorted part of the list grows from each end towards the middle. On each pass of the while loop, the largest item in the unsorted part is sorted to the top, and the smallest item is sorted to the bottom. The variables `bottom` and `top` represent the indices of the start and end of the unsorted part of the list.

Use the following function definitions to answer the questions on the following page.

```
def cocktailsort(lst):
    """ (list of number) -> NoneType

    Modify lst to sort the items from smallest to largest.

    >>> my_list = [4, 2, 5, 8, 6, 7, 3, 1]
    >>> cocktailsort(my_list)
    >>> my_list
    [1, 2, 3, 4, 5, 6, 7, 8]
    """

    top = len(lst) - 1
    bottom = 0

    while top > bottom:
        sort_to_top(lst, bottom, top)
        top = top - 1
        sort_to_bottom(lst, bottom, top)
        bottom = bottom + 1

def sort_to_top(lst, bottom, top):
    """ (list of number, int, int) -> NoneType

    Modify lst to swap the largest item in lst[bottom: top + 1] to index top.

    >>> my_list = [1, 2, 4, 6, 3, 5, 7, 8]
    >>> sort_to_top(my_list, 2, 5)
    >>> my_list
    [1, 2, 4, 5, 3, 6, 7, 8]
    """

    index_of_largest = bottom

    for j in range(bottom + 1, top + 1):
        if lst[j] > lst[index_of_largest]:
            index_of_largest = j

    lst[index_of_largest], lst[top] = lst[top], lst[index_of_largest]
```

Question continued on following page...

Question 9 continued...

Part (a) [4 MARKS] Write the function body for `sort_to_bottom` to complete the implementation. You can assume that each of the items in `lst` have a different value.

```
def sort_to_bottom(lst, bottom, top):
    """ (list of number, int, int) -> NoneType

    Modify lst to swap the smallest item in lst[bottom: top + 1] to index bottom.

    >>> my_list = [1, 2, 4, 5, 3, 6, 7, 8]
    >>> sort_to_bottom(my_list, 2, 4)
    >>> my_list
    [1, 2, 3, 5, 4, 6, 7, 8]
    """

    # complete the function body here
```

Part (b) [1 MARK]

Which of the following statements best describes the runtime behaviour of this sorting algorithm? Circle one.

(A) This algorithm does a different number of comparisons on a best case input vs a worst case input.

(B) This algorithm does not have a different best and worst case number of comparisons.

Question 11. [10 MARKS]

Two friends like to send messages to each other, but they do not want anyone else to read the messages.To keep the messages private, they change each letter in the original message to a different letter of the alphabet. For every letter, they both know which letter will be substituted for it, which is called an encoding. The message will contain only lowercase letters. It will not contain whitespace, digits, or punctuation.

The receiver of a secret message does not want to convert it back to the original message by hand. Instead, that person asks you to write a function to do it.

Following the function design recipe, define a function that given a secret message and the encoding used, returns the original message. Choose a meaningful function name, and add a docstring that includes the type contract, the description, and two examples that return different values. You must choose the types to use to represent the data, and you will be marked not only on the correctness of the function, but also on your choice of data structures.

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

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

Short Python function/method descriptions:

```
__builtins__:
  input([prompt]) -> 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.
  abs(x) -> number
    Return the absolute value of x.
  chr(i) -> Unicode character
    Return a Unicode string of one character with ordinal i; 0 <= i <= 0x10ffff.
  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 the list, tuple, dict, or string x.
  max(iterable) -> 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
  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[, mode]) -> file open for reading, writing, or appending
    Open a file.  Legal modes are "r" (read), "w" (write), and "a" (append).
  ord(c) -> integer
    Return the integer ordinal of a one-character string.
  print(value, ..., sep=' ', end='\n') -> NoneType
    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], 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 list starts at 0.  If step is not specified,
    the values are incremented by 1.

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
    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 of (object, object)
    Return the (key, value) pairs of D, as 2-tuples.
```

CONT'D...

```
file open for reading:
  F.close() -> NoneType
    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 of str
    Return a list of the lines from the file.  Each string retains any newline.

file open for writing:
  F.close() -> NoneType
    Close the file.
  F.write(x) -> int
    Write the string x to file F and return the number of characters written.

list:
  x in L --> bool
    Produce True if x is in L and False otherwise.
  L.append(x) -> NoneType
    Append x to the end of the 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.
  L.index(value) -> int
    Return the lowest index of value in L.
  L.insert(index, x) -> NoneType
    Insert x at position index.
  L.pop([index]) -> object
    Remove and return item at index (default last).
  L.remove(value) -> NoneType
    Remove the first occurrence of value from L.
  L.reverse() -> NoneType
    Reverse *IN PLACE*.
  L.sort() -> NoneType
    Sort the list in ascending order *IN PLACE*.

str:
  x in s --> bool
    Produce True if and only if x is in s.
  str(x) -> str
    Convert an object into its string representation, if possible.
  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.endswith(S2) -> bool
    Return True if and only if S ends with S2.
  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.index(sub) -> int
    Like find but raises an exception 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.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.lower() -> str
  Return a copy of the string S converted to lowercase.
S.lstrip([chars]) -> 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, new) -> str
  Return a copy of string S with all occurrences of the string old replaced
  with the string new.
S.rstrip([chars]) -> 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]) -> list of 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) -> bool
  Return True if and only if S starts with S2.
S.strip([chars]) -> 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.
```