

CSC 108H1 F 2011 Test 2
Duration — 45 minutes
Aids allowed: none

Student Number: _____

Last Name: _____ First Name: _____

Lecture Section: L0201

Instructors: Horton

*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 3 questions on 6 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.* Comments and docstrings are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code. No error checking is required: assume all user input and all argument values are valid.

1: _____/ 8

2: _____/ 6

3: _____/ 5

If you use any space for rough work, indicate clearly what you want marked.

TOTAL: _____/19

Question 1. [8 MARKS]

Each subquestion on this page has a small piece of code that is supposed to work as described in the comment statement but has a small part missing. For each one, add the missing part inside the box. Your solution must follow the instructions in the comment statement. Each subquestion is independent.

Part (a) [2 MARKS]

```
d = {'not': 'me'}
```

```
# Make changes to the dictionary so that the code below the box prints "all better now".
```

```
# Do not reassign d to a new dictionary.
```

```
for item in d.keys():  
    print item, d[item]
```

Part (b) [2 MARKS]

Provide the while loop condition in the box so that the program works as described in the comments.

```
campus = ['UTM', 'StG', 'UTSC']
```

```
tries = 0
```

```
favourite = raw_input("Which campus do you like best? ")
```

```
# give the user at most 3 tries to pick one of the valid choices
```

```
while
```

```
    print "you must pick from UTM, StG or UTSC"  
    favourite = raw_input("Pick again: ")  
    tries += 1  
if tries < 3:  
    print 'Yes ' + favourite + ' is great!'  
else:  
    print "Can't you follow instructions?"
```

In the box beside each piece of code on this page, write its output. If it would generate an error, say so, and give the reason for the error.

Part (c) [1 MARK]

```
temp = 18
L = [9, temp, 27]
temp = 99
print L
```

Part (d) [1 MARK]

```
temp = [5, 10, 15]
L = [temp, [3, 6]]
temp[1] = 99
print L
```

Part (e) [2 MARKS]

```
L = [[1], [2], [3]]
L2 = L[:]
for element in L:
    element.append(8)
print L
print L2
```

Question 2. [6 MARKS]

In Assignment 2, we defined an **association list** to be a list of lists, such as

```
[ [3, ["hello", 27.4, True]], ["drama", [13, "comedy", "goodbye", 1]] ].
```

Each sublist has two elements: the first is called a “key” and is a value of any type, and the second is a list of values (of any type) that are associated with that key. No key occurs more than once in an association list.

Part (a) [1 MARK]

Consider the following function.

```
def keys_for(alist, v):  
    '''Return a list of all the keys associated with value v in association list alist.'''
```

Write a call to the function that should return a list of length 2.

Part (b) [5 MARKS]

Now write the function. You do not need to repeat the `def` line or the docstring.

Question 3. [5 MARKS]**Part (a)** [4 MARKS]

As you know, a string can contain newline characters. When it does, we think of the string as having multiple lines within it.

Write the following function according to its docstring.

```
def line_lengths(s):  
    '''Return a list of ints containing the length of each line in string s. The newline  
    character at the end of a line does not count toward the line's length.'''
```

Part (b) [1 MARK]

Write a main block that will use your function to determine the lengths of the lines in file `poem.txt` and then print the list that your function produces.

Last Name: _____ First Name: _____

Short Python function/method descriptions:

`len(x)` -> integer
Return the length of the list or string x.

`sum(x)` -> integer
Return the sum of the elements in the list x.

`open(name[, mode])` -> file object
Open a file.

`range([start], stop, [step])` -> list of integers
Return a list containing the integers starting with start and ending with stop - 1 with step specifying the amount to increment (or decrement).

dict:

`D[k]` -> value
Return the value associated with the key k in D.

`k in d` -> boolean
Return True if k is a key in D and False otherwise.

`D.keys()` -> list of keys
Return the keys of D.

`D.values()` -> list of values
Return the values associated with the keys of D.

`D.items()` -> list of 2-tuples.
Return a list of D's (key, value) pairs.

`del D[k]`
Remove (key, value) pair with key k.

file (also called a "reader"):

`F.close()`
Close the file.

`F.read([size])` -> string
Read at most size bytes; with no size, read until EOF.

`F.readline([size])` -> string
Read next line, retaining newline; return empty string at EOF.

str:

`S.find(sub[,i])` -> integer
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.replace(old, new)` -> string
Return a copy of string S with all occurrences of the string old replaced with the string new.

`S.split([sep])` -> list of strings
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(prefix)` -> boolean
Return True if S starts with the specified prefix and False otherwise.

`S.strip()` --> string
Return a copy of S with leading and trailing whitespace removed.

list:

`L.append(x)`
Append x to the end of the list L.

`L.index(value)` -> integer
Return the lowest index of value in L.

`L.insert(index, x)`
Insert x at position index.