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

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

Last Name: _____ **First Name:** _____

Lecture Section: L0301 **Instructors: Craig**

---

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

1: _____/ 6
2: _____/ 6
3: _____/ 7

TOTAL: _____/19

Question 1. [6 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) [1 MARK]

```
d = {1:'remove me', 2:'take me away', 3:'KEEP'}
# Remove two items from d so that the code below the box prints 'KEEP'.
# Do not reassign d to a new dictionary.
```

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

Part (b) [1 MARK]

```
d = {}
# Add exactly one key-value pair to d so that the code below the box prints 'YES'.
```

```
for item in d:
    print item
```

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

Part (c) [2 MARKS]

```
L1 = ["once", "upon"]
L2 = L1
L1 = L1 + ['a', 'time']
print L1
print L2
```

Part (d) [2 MARKS]

```
L1 = ["Mary", "had", "a"]
L2 = L1
L2.append(["little", "lamb"])
print L1
print L2
```

Question 2. [6 MARKS]

Suppose we are keeping track of who is working with whom on a course assignment. We could represent the groups using a nested list of student numbers like this:
`[[2, 9], [4], [3, 1]]`. (Here we use one-digit student numbers to make the example easier to read.) In this example, we have two groups of two students, and one group of one student.

Part (a) [1 MARK]

Suppose we want to make sure that everyone is in a group, and no one is in more than one group. The following function checks this.

```
def valid_grouping(group_list, class_list):
    '''Return True if every student in class_list is in exactly 1 group
    according to group_list, and False otherwise.'''
```

Write a call to the function that should return `True` and involves a class list of 6 students.

Part (b) [5 MARKS]

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

Question 3. [7 MARKS]

Part (a) [5 MARKS]

Write the following function according to its docstring. The string matching that decides which lines to include in the result should be case-insensitive. However, the strings returned should be unmodified lines from the input file. For example, if the function is called with a file containing the line `Match\n`, and `s` has the value `maTch`, the original line (without any case changes) will be included in the list returned by the function.

```
def find_lines(f, s):
    '''Return a list of all lines in open file f that contain string s anywhere within them.'''
```

Part (b) [2 MARKS]

Write a main block that will use your function to print all the lines in file `poem.txt` that contain the string `love` (with any mixture of uppercase and lowercase letters.)

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.
```