CSC 108H1 F 2010 Test 2
Duration — 45 minutes
Aids allowed: none

Student Number: _____

Lab day, time, room: _____

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

Lecture Section: L5101 **Instructor: Daniel Zingaro**

---

*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 4 questions on 8 pages (including this one). *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. They may also get you part marks if you can't figure out how to write the code.
If you use any space for rough work, indicate clearly what you want marked.

1: _____/ 2
2: _____/ 6
3: _____/ 6
4: _____/ 6

TOTAL: _____/20

Question 1. [2 MARKS]

Give the output produced by each of the code fragments below. If a code fragment generates an error, say this, and also give us the reason for the error.

Part (a) [1 MARK]

```
L = ["Dan's", "favourite", "food", "is", "Kitkat"]
L = L[1:3]
print L
```

Part (b) [1 MARK]

```
s = "hellllo"
d = {}
for i in s:
    d[i] += 1
```

Question 2. [6 MARKS]

Diane has been known to spend hours and hours on end playing Tetris. However, she also enjoys spending as much time as possible helping students with CSC108 in office hours. But where does Diane spend more time?

To find out, Diane's colleagues have observed her behavior for a certain number of consecutive days and have created two parallel Python lists: a `tetris` list and an `office_hours` list. For day `i`, `tetris[i]` holds the number of hours spent on Tetris, and `office_hours[i]` holds the number of hours spent in office hours. Both lists are of equal length and contain only integers.

Write the following function according to its docstring.

```
def too_much_tetris (tetris, office_hours):
  '''tetris and office_hours are parallel lists of integers, as described above.
  Return True if Diane has spent at least one day
  where the number of tetris hours is more than the number of office hours.
  Return False otherwise.'''
```

Question 3. [6 MARKS]

Complete the following function according to its docstring description. **Hint**: `max(lst)` gives you the largest value in list `lst`.

```
def key_with_largest_value (d):
  '''d is a dict whose values are lists of integers.
  Return the key from d whose value contains the
  largest int among all of the values in d.
  For example, key_with_largest_value ({5:[2, 3], 4:[8,1]})
  returns 4. If multiple keys could be returned,
  return any one of them. d has at least one key.'''
```

Question 4. [6 MARKS]

Dan is interested in investigating a possible connection between whether students eat breakfast and their height. Dan's TA has spent hours at the Bahen entrance asking people for a `y` or `n` response to the question: "Do you eat breakfast?" For each of these students, the TA has also recorded their height in centimeters. The TA has produced a file where each data line contains a `y` or `n` followed by an integer number of centimeters (with no space between these two pieces of data). Unfortunately, the TA may have also added blank lines, and they could be anywhere. The file is guaranteed to include at least one data line for a student that eats breakfast and at least one data line for a student that does not. Here is a small sample file:

```
n140

y120


n150
y150
y175
```

Write the following function according to its docstring description.

```
def breakfast_averages (f):
  '''f is a file in the format described above. Return a tuple
  whose first component is the average height for students that
  eat breakfast, and whose second component is the average
  height for students that do not eat breakfast.'''
```

[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.]

Short Python function/method descriptions:

```
__builtins__:
  len(x) -> integer
    Return the length of the list or string x.
  max(L) -> value
    Return the largest value in L.
  open(name[, mode]) -> file object
    Open a file.
  range([start], stop, [step]) -> list of integers
    Return a list containing the integers starting with stop and ending witt  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] --> 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.
file (also called a "reader"):
  F.close(): Close the file.
  F.read([size]) -> read at most size bytes, returned as a string.
    If the size argument is negative or omitted, read until EOF is reached.
  F.readline([size]) -> next line from the file, as a string. Retain newline.
    A non-negative size argument limits the maximum number of bytes to return (an incomplete
    line may then be returned).  Return an empty string at EOF.
float:
  float(x) -> float
    Convert a string or number to a float, if possible.
list:
  x in L --> boolean
    Return True if x is in L and False otherwise.
  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.
  L.sort(): Sorts the list in ascending order.
int:
  int(x) -> integer
    Convert a string or number to an integer, if possible.  A floating point argument
    will be truncated towards zero.
```

Continued on reverse

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

```
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.index(sub [,start [,end]]) -> int
    Like S.find() but raise ValueError when the substring is not found.
  S.lower() -> string
    Return a copy of the string S converted to lowercase.
  S.lstrip([chars]) -> string
    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) --> string
    Return a copy of string S with all occurrences of the string old replaced with the string new.
  S.rstrip([chars]) -> string
    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 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) -> bool
    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.
  S.upper() -> string
    Return a copy of the string S converted to uppercase.
```