CSC 108H1 F 2009 Test 2
Duration — 35 minutes
Aids allowed: none

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

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

Lecture Section: L0101 **Instructor: 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 2 questions on 8 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: _____/10

2: _____/10

TOTAL: _____/20

Question 1. [10 MARKS]

For the purpose of this question, we will consider the letters"a", "e", "i", "o" and "u" (whether lowercase or uppercase) to be vowels, but not "y".

Part (a) [4 MARKS]

Consider the following function:

```
def vowels(s, i):
    '''Return the length of the longest sequence of consecutive vowels within
    s that starts at index i. s is a str of length at least one, and i is a
    valid index into s.'''
```

Complete the table below by adding four distinct test cases for function `vowels`. For each, provide specific values for `s` and `i`, the expected function result, and the purpose of the test case.

All of the test cases should be different from each other, and each should test something significant. You will receive no credit for repetitive test cases. Do not include tests that check for invalid input.

Value of `s`	Value of `i`	Expected result	Purpose of this test case

Part (b) [6 MARKS]

Write the function `vowels` according to its docstring. You must not use a for loop, and must not use `break`. Answers that do will receive no credit.

```
def vowels(s, i):
    '''Return the length of the longest sequence of consecutive vowels within
    s that starts at index i. s is a str of length at least one, and i is a
    valid index into s.'''
```

Question 2. [10 MARKS]

Part (a) [6 MARKS]

Write the following function according to its docstring:

```
def process_students(r):
    '''r is an open reader with data about students: their name, cdf account,
    age, college and home city. Each line has the following format:

    name,cdf,age,college,city

    There are no commas other than the ones used as separators.

    Return a dictionary in which each key is a college and its value
    is the list of cdf accounts for students at that college.'''
```

Part (b) [4 MARKS]

Write a program that opens a file called "students.txt" that is in the format described above, calls your function to build the dictionary, and pickles the dictionary to a file called "students.pck". You may assume that `cPickle` has been imported and that function `process_students` has been defined.

[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.
  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 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] --> 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.
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.lower() -> string
    Return a copy of the string S converted to lowercase.
  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) -> 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.
```

Continued on next page.

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

Short Python function/method descriptions, continued:

```
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.
  L.sort()
    Sorts the list in ascending order.
cPickle:
  dump(obj, file)
    Write an object in pickle format to the given file.
  load(file)
    Load a pickle from the given file
```