

PLEASE HAND IN

UNIVERSITY OF TORONTO  
Faculty of Arts and Science  
APRIL/MAY 2009 EXAMINATIONS

PLEASE HAND IN

CSC 108H1S  
Instructor: Horton  
Duration — 3 hours  
Examination Aids: None

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 fill out the identification section above,  
and read the instructions below carefully.)*

---

MARKING GUIDE

This final examination consists of 10 questions on 21 pages (including this one). When you receive the signal to start, please make sure that your copy of the examination 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.

You do not need to put import statements in your answers.

You may not use break or continue on this exam.

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

Assume all input is valid unless otherwise indicated; there is no need to error-check.

# 1: \_\_\_\_/ 8

# 2: \_\_\_\_/ 6

# 3: \_\_\_\_/ 7

# 4: \_\_\_\_/12

# 5: \_\_\_\_/12

# 6: \_\_\_\_/ 6

# 7: \_\_\_\_/10

# 8: \_\_\_\_/12

# 9: \_\_\_\_/12

# 10: \_\_\_\_/ 8

TOTAL: \_\_\_\_/93

*Good Luck!*

**Question 1.** [8 MARKS]

Complete the following function according to its docstring description.

```
def clump(L, k):  
    '''Return a list that is the same as list L, but with the items grouped into  
    sublists of length k (except for the last sublist, which may be shorter).  
    k must be >= 1.  
    Example: clump([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 4) returns the list  
    [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10]].'''
```

**Question 2.** [6 MARKS]

Complete the following function according to its docstring description.

Hint: `media.get_pixel` lets you get the pixel at desired coordinates from a picture. You can then either get or set the pixel's colour.

```
def cutout(p1, p2, x, y, h, w):  
    '''p1 and p2 are Pictures with the same height and width. Change p1 so  
    that every pixel in a certain rectangle has the same colour as the  
    corresponding pixel in p2. The rectangle to change has its upper-left  
    corner at pixel (x, y), and has height h and width w.  
    The entire rectangle falls within the borders of p1 (and therefore within  
    the borders of p2, since they have the same dimensions).'''
```

**Question 3.** [7 MARKS]

Read the help for classes Student and Course below.

```
class Student
```

```
| A student with a surname, first name, student ID, and a set of courses  
| taken, each with an integer grade.  
|  
| Methods defined here:  
|  
| __init__(self, surname, first_name, ID)  
|     A new student with the given surname and first_name (strings), and  
|     ID (an int), and with no courses completed.  
|  
| average(self)  
|     Return this Student's average course grade, or -1 if they have  
|     completed no courses.  
|  
| complete(self, c, g)  
|     Record that this Student completed the course named c (a string)  
|     with grade g (an int). If they have taken course c before,  
|     replace their old grade with the new one.
```

```
class Course
```

```
| A course offering with a course name, term, a set of students who are  
| enrolled, and an enrolment limit.  
|  
| Methods defined here:  
|  
| __init__(self, name, t, m)  
|     A new Course called name (a string), offered in term t (a string),  
|     with no students enrolled, and a maximum enrolment of m (an int).  
|  
| drop(self, s)  
|     Drop Student s from this course. If there is no such student  
|     in the course, do nothing.  
|  
| enrol(self, s)  
|     If this course is not already at its maximum enrolment,  
|     enrol Student s in this course.
```

**Part (a)** [5 MARKS] Complete the code below, according to the comments.

```
# Create two Students, Diane Horton (with student number 9912) and  
# Jane Davies (with student number 9934).
```

```
# Create a course "csc108", offered in term "2009 winter" with an  
# enrolment limit of 100.
```

```
# Enrol Jane and Diane in csc108.
```

```
# Drop Diane from csc108, and give Jane a grade of 81.
```

**Part (b)** [2 MARKS] Suppose we wanted to be able to use the built-in function `list.sort` in order to sort a list of Students according to their average grade. What method would we have to define in order for this to work, and where would we have to put it?

**Question 4.** [12 MARKS]

A palindrome is a string that reads the same forwards and backwards. Examples of palindromes include "lol", "abba", "radar", and "pickle elkcip".

**Part (a)** [8 MARKS] For each palindrome function below, indicate whether or not it works under all circumstances described in the following docstring: `'''Return True if string s is a palindrome and return False otherwise.'''`

Don't guess. There is a 1-mark deduction for wrong answers on this question.

```
def palindrome1(s):
```

```
    n = len(s)
    pal = True
    for i in range(n/2):
        if s[i] != s[n-i-1]:
            pal = False
    return pal
```

Circle one:          works          doesn't work

```
def palindrome2(s):
```

```
    n = len(s)
    pal = True
    for i in range(n/2):
        if s[i] == s[n-i-1]:
            pal = True
        else:
            pal = False
    return pal
```

Circle one:          works          doesn't work

```
def palindrome3(s):
```

```
    n = len(s)
    pal = True
    i = 0
    while i < n/2 and pal:
        if s[i] == s[n-i-1]:
            i = i + 1
        else:
            pal = False
    return pal
```

Circle one:          works          doesn't work

```
def palindrome4(s):  
    n = len(s)  
    pal = True  
    i = 0  
    while i < n and pal:  
        if s[i] == s[n-i-1]:  
            i = i + 1  
        else:  
            pal = False  
    return pal
```

Circle one:        works        doesn't work

**Part (b)** [4 MARKS] For each function that doesn't work, write a nose test that demonstrates one case where it fails. You may assume that all necessary imports have been done and that there is a suitable `if __name__ == "__main__"` section; just write the function(s).

**Question 5.** [12 MARKS]

When you use our website to declare your partner for a csc108 assignment, a record of that declaration is stored in a file. The function `pair_up` processes the partner file.

```
def pair_up(f):
    '''f is an open file that contains lines of the form:
    ACCOUNT1 ACCOUNT2
    Return a dictionary in which each key is a student's account and each
    value is the account of that student's partner. If A is the partner of B,
    then B is the partner of A, so the dictionary stores two key-value pairs
    for each partnership.
    If a student makes more than one declaration with the same partner, print
    a warning message for each extra declaration. And if a student makes
    more than one declaration and they involve different partners, print
    a warning message for each extra declaration.'''
```

As an example, this file:

```
c9horton c9davies
c7dickin c8steven
c8fairgr c8zemelr
c8steven c7dickin
c6mcilra c7fleete
c8brecht c6mcilra
c8pitass c8zemelr
```

would return the following dictionary of partners:

```
{'c9horton': 'c9davies', 'c7fleete': 'c6mcilra', 'c8fairgr': 'c8zemelr', \
 'c9davies': 'c9horton', 'c8steven': 'c7dickin', 'c7dickin': 'c8steven', \
 'c6mcilra': 'c7fleete', 'c8zemelr': 'c8fairgr'}
```

and also print these warning messages:

```
c8steven c7dickin : declared partnership more than once
c6mcilra : declared more than one partner
c8zemelr : declared more than one partner
```

On the next page, write the body of function `pair_up`. You do not need to write out the docstring. Use the same content and format for your warning messages as shown above.



```
def pair_up(f):
```

**Question 6.** [6 MARKS]

Don't guess. There is a 1-mark **deduction** for wrong answers on this question.

**Part (a)** [2 MARKS]

```
def silly1(L):  
    for i in range(len(L)):  
        print L[i]  
  
    for i in range(len(L) - 1, 0, -1):  
        print L[i]
```

Let  $n$  be the size of the list passed to this function. Which of the following most accurately describes how the runtime of this function grow as  $n$  grows? Circle one.

- (a) It grows linearly, like  $n$  does.    (b) It grows quadratically, like  $n^2$  does.  
(c) It grows less than linearly.    (d) It grows more than quadratically.

**Part (b)** [2 MARKS]

```
def silly2(L):  
    for i in range(300):  
        for j in range(len(L)):  
            L[j] = L[j] + i
```

Let  $n$  be the size of the list passed to this function. Which of the following most accurately describes how the runtime of this function grow as  $n$  grows? Circle one.

- (a) It grows linearly, like  $n$  does.    (b) It grows quadratically, like  $n^2$  does.  
(c) It grows less than linearly.    (d) It grows more than quadratically.

**Part (c)** [2 MARKS]

```
def silly3(L):  
    for i in range(len(L)):  
        for j in range(len(L)):  
            print L[i], L[j]
```

Let  $n$  be the size of the list passed to this function. Which of the following most accurately describes how the runtime of this function grow as  $n$  grows? Circle one.

- (a) It grows linearly, like  $n$  does.    (b) It grows quadratically, like  $n^2$  does.  
(c) It grows less than linearly.    (d) It grows more than quadratically.

**Question 7.** [10 MARKS]

Read the docstring for function `intersect_dictionaries`.

```
def intersect_dictionaries(d1, d2):  
    '''d1 and d2 are dictionaries. Return a new dictionary  
    whose keys are all those things that are keys in BOTH d1 and d2.  
    A key's value in the new dict is the sum of its value in d1 and  
    its value in d2.'''
```

**Part (a)** [4 MARKS] For each of the following values for `d1` and `d2`, what is the return value of the function?

d1	d2	return value
{9: 4, 2: 5, 3: 17}	{2: 2, 3: 1, 9: 4}	
{9: 4, 2: 5, 3: 17}	{2: 2, 1: 13, 9: 4, 7: 7}	
{}	{2: 2, 1: 13, 9: 4, 7: 7}	
{}	{}	

**Part (b)** [6 MARKS] Write the body of function `intersect_dictionaries`.

**Question 8.** [12 MARKS]

Below is the beginning of a very simple program that lets the user update the quantity of an inventory item. For example, if the user enters item "rice" and quantity 7, and then clicks "record", the value for key "rice" in the inventory dictionary should increase from its initial value of 15 to 22. A negative quantity should decrease the value in the dictionary.

A label is used to report back on the status of the update. It should either say "New item" or "New quantity", followed by the updated quantity. In our example, the status label would say "New quantity 22".

```
from Tkinter import *

# Definitions of any helper function(s) would go here.

if __name__ == "__main__":

    inventory = {"peas": 3, "rice": 15, "peanut butter": 8}

    window = Tk()
    frame = Frame(window)
    frame.pack()

    label = Label(frame, text="Item:")
    label.pack()

    entry1 = Entry(frame)
    entry1.pack()

    label2 = Label(frame, text="Quantity:")
    label2.pack()

    entry2 = Entry(frame)
    entry2.pack()

    # The following statement is incomplete:
    record_command = ??
    record_button = Button(frame, text="Record", command=record_command)
    record_button.pack()

    status = StringVar()
    label3 = Label(frame, textvariable=status)
    label3.pack()

    window.mainloop()
```

**Part (a)** [3 MARKS] The statement that sets the value for `record_command` is incomplete. Write a suitable statement here.

```
record_command =
```

**Part (b)** [9 MARKS] Write any helper function(s) you need here. You may assume that the user's `input` is valid. You may not access any global variables in your function(s); all information must flow through parameters and/or return values.

**Question 9.** [12 MARKS]

**Part (a)** [6 MARKS] Complete the following function according to its docstring description. You must use a while loop. Remember that `break` and `continue` are not allowed on this exam.

```
def run_length(L, i):  
    '''L is a list of ints, and 0 <= i < len(L).  
    Return the length of the sequence of consecutive equal ints starting at  
    position i in L. For example, run_length([7, 8, 8, 8, 4], 1) is 3 and  
    run_length([9, 9, 3, 5, 1, 1], 2) is 1.'''
```

**Part (b)** [6 MARKS] Use function `run_length` as a helper to write function `run_length_encode` according to its docstring description. Again, you must use a while loop, and remember that `break` and `continue` are not allowed on this exam.

You can get full marks for this part even if your answer to part (a) is wrong or missing.

```
def run_length_encode(L):
    '''L is a list of ints.
    Return a new list that replaces each sequence of consecutive equal ints
    with two things: the int, and the number of times it occurred in a row.
    For example, if called with the list
    [7, 4, 4, 4, 4, 8, 8, 4, 1, 1, 1, 1, 1], then return the list
    [7, 1, 4, 4, 8, 2, 4, 1, 1, 5].
    Notice that even sequences of length 1 are replaced.'''
```

**Question 10.** [8 MARKS]

The function in the program below does not meet its specifications.

```
def remove(L, n):  
    '''Remove all occurrences of int n from list of ints L.  Change n to be  
    the number of occurrences that were removed.'''  
  
    count = 0  
    while n in L:  
        L.remove(n)  
        count = count + 1  
  
    n = count  
  
if __name__ == "__main__":  
    num_list = [5, 7, 9, 2, 3, 9, 9, 7, 1, 7]  
    num = 7  
  
    # Remove all occurrences of 7 from num_list.  
    remove(num_list, num)  
  
    # Print the revised list and the number of occurrences that were removed.  
    print "Revised list is", num_list  
  
    print "Number of occurrences removed is", num
```

**Part (a)** [1 MARK] What would the output of this program be if the function `remove` met its specifications?

**Part (b)** [2 MARKS] What is the actual output of this program?

**Part (c)** [2 MARKS] It is, in fact, impossible for the function to meet its specifications. Why?

**Part (d)** [3 MARKS] Make the minimum changes needed so that the program does what its comments say. Change function `remove` and its docstring as needed. Make your changes directly on the code **above**.



*[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.]*

*[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\_\_:**

**lambda:** expr -> function

Returns a function that evaluates the Python expression expr.

**len(x)** -> integer

Return the length of the list, tuple, dict, or string x.

**max(L)** -> value

Return the largest value in L.

**min(L)** -> value

Return the smallest value in L.

**open(name[, mode])** -> file object

Open a file. Legal modes are "r" (read), "w" (write), and "a" (append).

**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).

If start is not specified, the list starts at 0. If step is not specified, the values are incremented by 1.

**Color:**

**black**

RGB: 0, 0, 0

**Color(red, green, blue)** -> Color

Define a Color with the specified red, green, and blue components.

**white**

RGB: 255, 255, 255

**dict:**

**D.get(k)** -> value

Return the value associated with the key k in D.

**D.has\_key(k)** -> 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 (key, value) pairs

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

**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 (End of File) 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 be returned then). Return an empty string at EOF.

**float:**

**float(x)** -> floating point number

Convert a string or number to a floating point number, if possible.

**int:**

**int(x)** -> integer

Convert a string or number to an integer, if possible. A floating point argument will be truncated towards zero.

**list:**

**L.append(x)**

Append x to the end of the list L.

`L.index(value) -> integer`  
 Returns the lowest index of value in L.  
`L.insert(index, x)`  
 Insert x at position index.  
`L.remove(value)`  
 Removes the first occurrence of value from L.  
`L.sort()`  
 Sorts the list in ascending order.

`media:`  
`choose_file() -> str`  
 Prompt user to pick a file. Return the path to that file.  
`create_picture(int, int) -> Picture`  
 Given a width and a height, return a Picture with that width and height. All pixels are white.  
`get_blue(Pixel) -> int`  
 Return the blue value of the given Pixel.  
`get_color(Pixel) -> Color`  
 Return the Color object with the given Pixel's RGB values.  
`get_green(Pixel) -> int`  
 Return the green value of the given Pixel.  
`get_pixel(Picture, int, int) -> Pixel`  
 Given x and y coordinates, return the Pixel at (x, y) in the given Picture.  
`get_red(Pixel) -> int`  
 Return the red value of the given Pixel.  
`load_picture(str) -> Picture`  
 Return a Picture object from file with the given filename.  
`set_blue(Pixel, int)`  
 Set the blue value of the given Pixel to the given int value.  
`set_color(Pixel, Color)`  
 Set the RGB values of the given Pixel to those of the given Color.  
`set_green(Pixel, int)`  
 Set the green value of the given Pixel to the given int value.  
`set_red(Pixel, int)`  
 Set the red value of the given Pixel to the given int value.  
`show(Picture)`  
 Display the given Picture.

`str:`  
`str(x) -> string`  
 Convert an object into its string representation, if possible.  
`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) -> integer`  
 Like find but raises an exception if sub does not occur in S.  
`S.isdigit() -> boolean`  
 Return True if all characters in S are digits and False otherwise.  
`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.strip() -> string

Return a copy of S with leading and trailing whitespace removed.

Tkinter:

Button:

Button(parent, [text=],[textvariable=],[command=]) -> button object

Construct a button with the given parent.

Entry:

Entry(parent) -> entry object

Construct an entry field with the given parent.

E.get() -> string

Return the text in the entry field E.

Frame:

Frame(parent) -> frame object

Construct a frame widget with the given parent.

Label:

Label(parent, [text=],[textvariable=]) -> label object

Construct a label with the given parent.

TkVariables:

DoubleVar() -> Tk float variable object

Construct a Tkinter float variable.

IntVar() -> Tk int variable object

Construct a Tk int variable.

StringVar() -> Tk string variable object

Construct a Tk string variable.

tkvar.get() -> int

Return the value of tkvar.

tkvar.set(value)

Set tkvar to value.

Window:

Tk() -> window

Return a new window widget.

All Widgets:

A.grid(row=r, column=c)

Place widget A in row r and column c.

A.pack()

Place widget A below the last widget that was packed.

