PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

AUGUST 2010 EXAMINATIONS

CSC 108 H1Y
Instructors: Freund

Duration — 3 hours

Examination Aids: None

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

Family Name(s): _____

Given Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully.*

This final examination paper consists of 10 questions on 22 pages (including this one). *When you receive the signal to start, please make sure that your copy of the final 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.
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: _____/ 6
2: _____/ 8
3: _____/ 7
4: _____/12
5: _____/ 8
6: _____/18
7: _____/ 7
8: _____/ 6
9: _____/10
10: _____/ 8

TOTAL: _____/90

Good Luck!

CONT'D...

Question 1. [6 MARKS]

Part (a) [3 MARKS] Write the following function, according to its docstring:

```
def capitalize_items_1(lst):
    '''Change each string in lst to be all upper-case.
    lst is a list of strings. This function does not return a value.'''
```

Part (b) [3 MARKS] Write the following function, according to its docstring:

```
def capitalize_items_2(lst):
    '''Return a new list with the same items as lst, only in upper-case.
    lst is a list of strings. This function does not change lst.'''
```

Question 2. [8 MARKS]

Write the following function, according to its docstring:

```
def substitute_digits(s, subst):
    '''
    s is a string and subst is a string of length 10.
    Return a copy of s, where each digit in s is replaced by the corresponding
    character from subst.
    That is, the character '0' is replaced by the first character of subst,
    '1' is replaced by the second character of subst, ..., and '9' is replaced
    by the last character of subst.
    For example: The call substitute_digits('Hello 43212', 'xrehTlmnop')
    should return the string 'Hello There'.
    '''
```

Question 3. [7 MARKS]

The left-hand column in the table below shows a series of code fragments to be interpreted by the Python shell. For each, show the expected output in the right-hand column; if it would generate an error say so, and give the reason why.

Hint: Use the memory model to predict what will happen. The **next page is provided for your rough work.**

Code	Output (or "error" plus reason)

`x = [(1,2), ['a','b'], ([1,2], (3,4))]`
`for item in x:`
`  print len(item)`	

`x = [(1,2), ['a','b'], ([1,2], (3,4))]`
`for i in x:`
`  i = 0`
`print x`	

`x = [(1,2), ['a','b'], ([1,2], (3,4))]`
`for i in range(len(x)):`
`  x[i] = 0`
`print x`	

`x = [(1,2), ['a','b'], ([1,2], (3,4))]`
`for item in x:`
`  item[0] = 0`
`print x`	

`x = [(1,2), ['a','b'], ([1,2], (3,4))]`
`x[1][0] = 'c'`
`print x`	

`x = [(1,2), ['a','b'], ([1,2], (3,4))]`
`x[2][0] = 'c'`
`print x`	

`x = [(1,2), ['a','b'], ([1,2], (3,4))]`
`x[2][0][0] = 'c'`
`print x`	

Use the space below for rough work. This page will not be marked.

Question 4. [12 MARKS]

Below are several functions that attempt to return True if the given string is a palindrome, and False otherwise. Recall that a palindrome is any string that doesn't change when you reverse it. For example, the following strings are palindromes: '', 'a', 'aa', 'aba', 'abccba', 'abcba'. The following strings are not palindromes: 'ab', 'ba', 'abcdba', 'aaaab'.

For each version below, indicate whether or not the function works. If it does not, write a call to the function that demonstrates one case where it fails (*i.e.*, the return value is incorrect or the function generates an error), and state what happens.

Part (a) [3 MARKS]

```
def is_palindrome1(s):
    answer = True
    while len(s) > 1:
        if s[0] == s[-1]:
            answer = True
        else:
            answer = False
        s = s[1:-1]
    return answer
```

Does the function work (circle one)?: works doesn't work

If it doesn't work, a call to the function that demonstrates this:

and the outcome of that call (circle one): incorrect return value error

Part (b) [3 MARKS]

```
def is_palindrome2(s):
    if s == '':
        return True
    i = 0
    j = len(s) - 1
    while i < j:
        if s[i] != s[j]:
            return False
        i = i + 1
        j = j - 1
    return True
```

Does the function work (circle one)?: works doesn't work

If it doesn't work, a call to the function that demonstrates this:

and the outcome of that call (circle one): incorrect return value error

Part (c) [3 MARKS]

```
def is_palindrome3(s):
    for i in s:
        if s[i] != s[-1 - i]:
            return False
    return True
```

Does the function work (circle one)?: works doesn't work

If it doesn't work, a call to the function that demonstrates this:

and the outcome of that call (circle one): incorrect return value error

Part (d) [3 MARKS]

```
def is_palindrome4(s):
    for i in range(len(s)):
        if s[i] != s[-1 - i]:
            return False
    return True
```

Does the function work (circle one)?: works doesn't work

If it doesn't work, a call to the function that demonstrates this:

and the outcome of that call (circle one): incorrect return value error

Question 5. [8 MARKS]

Suppose we record the marking information for this course in a dictionary, where each key (a string) is the name of an exercise (e.g. 'A1', 'test2', 'codelab', etc.), and its value is a dictionary with the following format:

```
{"out_of": 22,
 "marks": {"c0student1" : 19,
           "c0student2" : 15,
           # More entries mapping cdf names to marks ...
           }
}
```

Your goal is to print a summary report of the marking information.
The report should have a line for each exercise in the dictionary. Each line should contain the name of the exercise and the average, maximum and minimum marks.

Part (a) [6 MARKS] Write the function on the next page according to its docstring. Before you do that, you may want to read part (b) of this question.

Part (b) [2 MARKS] You can earn **two bonus marks** by satisfying the following requirements (one mark per requirement):

1. Marks are printed in percentage (i.e. out of 100).
2. Marks are printed with exactly one digit to the right of the decimal point.

Here is an example of a line of output that your function may produce:

```
A1, average: 80.3, min: 46.3, max: 100.0.
```

```
def print_summary_report(data):
    '''Print a summary report of the marks in data, as described on the previous page.
    data is a dictionary, as described on the previous page.'''
```

Question 6. [18 MARKS]

For this question, you will write a function that reads text files containing marking information, and organizes the data in a dictionary (like the one used in the previous question). Here is an example of a file your program should be able to process:

```
A1,Test1,A2
36,22,56
c0stdnt1,15,,27
c0stdnt2,34,19,52
```

More generally, your function should be able to process files of the following format:

- The first line contains exercise names, separated by commas.
- The second line indicates what each exercise is out of.
- The remaining lines specify the marks that each student got, one student per line.
  Entries on the line are separated by commas. The first entry is the cdf username of the student, and the remaining entries are marks. Each mark is either an integer or is empty (e.g. see the line that starts with `c0stdnt1` in the example above).

Your function should return a dictionary with the following format:

- The keys of the dictionary are the names of the exercises (as strings).
- The values of the dictionary are dictionaries with two keys, `'out_of'` and `'marks'`.
  The value of `'out_of'` is an int (indicating what the exercise is out of). The value of `'marks'` is a dictionary mapping cdf usernames (strings) to marks (ints).

For example, if your function is called on the file above, it should return the following dictionary:

```
{'A1':    {'out_of': 36, 'marks': {'c0stdnt1': 15, 'c0stdnt2': 34}},
 'Test1': {'out_of': 22, 'marks': {'c0stdnt2': 19}},
 'A2':    {'out_of': 56, 'marks': {'c0stdnt1': 27, 'c0stdnt2': 52}}
}
```

Notice that, since c0stdnt1 does not have a mark for Test1, the marks dictionary of Test1 does not contain the key c0stdnt1. Your program should handle missing marks the same way - Do not create entries for them.

Part (a) [15 MARKS] Write the function on the next page, according to its docstring.
You can assume that the input file to your function follows the described format exactly, and that the number of entries on the first line, the number of entries on the second line and the number of mark entries in each of the remaining lines are all the same.

Suggestion: Think about how to process the first two lines. After you are done that, you can start thinking about processing the remaining lines.

```
def process_marks_file(filename):
    '''Return a dictionary that represents the data in filename, as described
    on the previous page. filename (string) is the name of a file containing
    marks in the format describes on the previous page. '''
```

Part (b) [3 MARKS] Write the main block (including its `if` statement) of a program that reads a marks file called `marks.txt` and prints a summary report.
You can assume that both functions, `process_marks_file` and `print_summary_report` (as defined in this question and the previous one), are available and work correctly.

Question 7. [7 MARKS]

On the next page is the outline of a program that allows you to watch TV on your computer. Its main block will use a variable called `data` to store usage information (e.g. movie orders, channel subscriptions, etc). You have three tasks:

1. Complete the main loop so that it calls the appropriate function from the `main_choices` dictionary, and exits when the user clicks *Cancel*. Note that all these functions need one argument: `data`.

2. Modify the program as appropriate so that it initializes the variable `data` by loading it from a cPickle file called `data.pkl`, if that file exists. If not, it should initialize `data` to be an empty dictionary. You can use `os.path.isfile('data.pkl')` to see if the file `data.pkl` exists (this call returns True if the file exists, and False otherwise).

3. Modify the program so that before it exits, it saves the contents of variable `data` in a cPickle file called `data.pkl`.

You may add, delete or change existing code. If your new code won't fit in the space available, use arrows to point to where the pieces should go.

```
import easygui, cPickle, sys, os
# Definitions of functions view_channel, order_movie, order_channel and
# cancel_channel omitted to save space.


if __name__ == '__main__':




    main_choices = {"View Channel" : view_channel, "Order Movie" : order_movie,
                    "Order Channel": order_channel, "Cancel Channel": cancel_channel}

    choice = easygui.buttonbox("Action", "Python TV", main_choices.keys())

    while choice != None:

        # TODO: Call a function based on choice ...




        # When the function is done, display the main choice-box again.
        choice = easygui.buttonbox("Action", "Python TV", main_choices.keys())
```

Question 8. [6 MARKS]

This question is made up of three multiple answer questions. **Do not guess** the answer. There is a 1-mark **deduction** for wrong answers on this question.

Part (a) [2 MARKS]

```
def f1(lst):
    sum = 0
    for i in range(2, 1003):
        # Assume that getting a random item from a list takes 1 step.
        x = lst[random.randint(0, len(lst) - 1)]
        sum += x
    return sum
```

Let n be the size of the list `lst` 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 f2(lst):
    result = []
    i = 1
    while i < len(lst):
        # Assume that appending an item to a list takes 1 step.
        result.append(lst[i])
        i = i * 3
    return result
```

Let n be the size of the list `lst` 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 f3(lst):
    smallest = 1000
    for x in lst:
        if x < smallest:
            smallest = x
    largest = -1
    for x in lst:
        if x > largest:
            largest = x
    return smallest, largest
```

Let n be the size of the list `lst` 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 9. [10 MARKS]

Consider the following class:

```
class Computer
 |  A class representing a computer.
 |
 |  Methods defined here:
 |
 |  __init__(self, model, price)
 |      A new computer with the given model (string) and price (either float or int).
 |
 |  __cmp__(self, other)
 |      Return -1 if this is computer is cheaper than other, +1 if other is cheaper,
 |      and 0 if this computer and other have the same price.
 |
 |  __str__(self)
 |      Return a string containing the model and price of this computer.
 |
 |  get_model(self)
 |      Return the model of this computer as a string.
 |
 |  add_opt_accessory(self, accessory)
 |      Add accessory as an optional accessory for this computer.
 |      accessory is an Accessory object (the Accessory class is defined somewhere else).
 |
 |  get_accessories(self):
 |      Return optional accessories for this computer as a list of Accessory objects.
 |
```

Part (a) [7 MARKS]

1. Create two `Computer` instances, a *Thinkpad100* that costs 500$ and a *Vaio350* that costs 650$, and make the variables `thinkpad` and `vaio` refer to these instances, respectively.

2. Which method does the above code call without mentioning it by name?

3. What would the call `thinkpad < vaio` return?

4. Which method does `thinkpad < vaio` call without mentioning it by name?

5. Suppose we create another Computer, *HP777*, which costs 650$, and make a variable called `hp` refer to it. What would the call `vaio == hp` return?

6. Write a loop that prints the optional accessories of `hp`, one per line (assume that Accessory objects have their own `__str__` method that returns a string representation of an accessory).

Part (b) [3 MARKS] Complete the following function according to its docstring description.

```
def cheapest(computers):
    '''computers is a non-empty list of Computer objects.
    Return the model (as a string) of the cheapest computer in computers.'''
```

Question 10. [8 MARKS]

Throughout this question, assume that we are sorting lists into non-descending order (from smallest to largest). **Do not guess. There is a one-mark deduction for incorrect answers.**

Part (a) [2 MARKS]

Consider the following lists:

```
lst1 = [7, 6, 5, 4, 3, 2, 1]
lst2 = [6, 1, 2, 3, 7, 4, 5]
```

Which list would cause bubblesort to do less swaps? Circle one answer.

`lst1`
`lst2`
They would both require an equal number of swaps

Part (b) [2 MARKS]

What is the maximum possible number of swaps that bubble sort performs on a list with 5 elements? Give an example of such a list.

Part (c) [2 MARKS]

Consider a list of 100 distinct numbers in order from largest to smallest, in other words, backwards to the order we want. Which sorting technique would cause items to be moved more times, if called on this list? Circle one.

Bubble sort Selection sort

Part (d) [2 MARKS]

We are partly through sorting a list, and have completed 3 passes through the data. The list currently contains

`[13, 25, 41, 47, 33, 1, 62, 2, 99, 10, 60]`

Which sorting technique are we using? Circle one.

Bubble sort Selection sort Insertion sort

[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__:
  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.
cPickle:
  dump(obj, file)
    Write an object in pickle format to the given file.
  load(file) --> object
    Load a pickle from the given file
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.get(k) -> value
    Return D[k] if k in D, otherwise return None.
  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:
  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
    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.reverse()
    Reverse *IN PLACE*
  L.sort()
    Sorts the list in ascending order.
str:
  x in s --> boolean
    Return True if x is in s and False otherwise.
  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.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.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.
```

Total Marks = 90