CISC-121
Quiz #2
March 15, 2013

Do not turn this page until the quiz officially begins.

STUDENT NUMBER ____ ____ ____ ____ ____ ____ ____ ____

Please do not write your name anywhere on this quiz. I recommend writing your student number at the top of each page.

You are expected to behave considerately towards your fellow students. Noisy or disruptive behaviour will not be tolerated. **Turn your phone off.** If you finish early, you may quietly leave the room after handing in your quiz. **However, nobody will be permitted to leave during the last 10 minutes of the quiz.**

If you leave the quiz early, do not stand around outside the exam room talking. This is extremely distracting for students still working.

Academic dishonesty will not be tolerated. Keep your eyes on your own paper or the blackboard at the front of the room. You may bring one sheet of paper to the quiz. You may not refer to other notes or books during the quiz. You may not use a calculator or computer during the quiz.

Please try to write your answers in the space provided. If you need to write your answers elsewhere, please indicate clearly where on the quiz your answers are to be found.

You do not need to add comments to your code.

You have 50 **minutes** to complete the quiz. Good luck!

Question 1	/10
Question 2	/10
Question 3	/10
Question 4	/10
Question 5	/10
Total	/50

Question 1 (10 Marks):

In Python the elements of a list can be lists, and the embedded lists do not need to all have the same length. For example, we might have

A = [[1,2,3,4],[2,7],[9,1,14]]

The list A consists of 3 embedded lists, with length 4, 2 and 3.

Suppose for some application it is essential that the lists in a list of lists must all have the same length. There are two simple solutions: trim all the lists to the length of the shortest list by deleting values from the end of the longer lists, or pad all the lists up to the length of the longest list by adding 0's at the end of each list that is too short.

Either write a Python function called **trim** or write a Python function called **pad.** Whichever you choose, it must take a single parameter (the list of lists) and return a copy of the list, trimmed or padded (depending on your choice). For example, using the list A shown above,

trim(A) would return [[1,2],[2,7],[9,1]]
pad(A) would return [[1,2,3,4],[2,7,0,0],[9,1,14,0]]

Your function does not need to check to see if the object passed in to the parameter is a list of lists – you can assume that this check is carried out in the main program before the function is called.

Write your answer in the box on the next page:

Write your function **trim** or **pad** in this box

```
def trim(A):
     ml = len(A[0])
     for r in A:
          if len(r) < ml:
               ml = len(r)
     new_A = []
     for r in A:
          new_A.append(r[:ml])
     return new_A
```

```
def pad(A):
     ml = len(A[0])
     for r in A:
          if len(r) > ml:
               ml = len(r)
     new_A = []
     for r in A:
          new_r = r[:]
          new_r.extend([0]*(ml-len(r)))
          new_A.append(new_r)
     return new_A
```

Marking: If their answer shows that they had the right idea they should get at least 5/10, even if they have errors. Minor syntax errors (missing : etc) can be ignored. Forgetting that append() does not return a list (so x = x.append(y) doesn't work) should cost 1 mark.

Their solutions do not have to be efficient or concise.

Question 2 (10 Marks):

Here is a recursive Python function

```
def rec_fun(x,y):
     print "start ",x,y

     if (x <= 1) or (y <= 1):
          sum = x + y
     else:
          sum = rec_fun(x-1,y) + rec_fun(x, y-1)

     print sum
     return sum
```

Show the output that results when the function is called by the following line:

```
n = rec_fun(3,2)
```

```
start  3 2
start  2 2
start  1 2
3
start  2 1
3
6
start  3 1
4
10
```

Marking: basically 1 mark for each line, but if it is clear that they made an early mistake that threw everything else off, please take that into consideration. As with Q1, if their answer shows that they have the right idea they should be getting at least 5/10

Question 3 (10 Marks):

Suppose the function **Molasses(n)** returns a value. We need to compute the function's result for each value in a list of numbers, some of which may repeated. For example, the list of numbers for which Molasses(n) is to be evaluated might be [7, 10000, -35, 7, -35, 7]. Notice that 7 occurs three times in this list. In this case, we need to compute Molasses(7), Molasses(10000), Molasses(-35), Molasses(7), Molasses(-35) and Molasses(7).

The function Molasses(n) takes such a long time to compute its result that we never want to repeat a call to Molasses(n) if we can avoid it.

What feature of Python would you use to minimize the number of times Molasses(n) must be called to obtain its result for each value in the list? Why? (Hint: you may want to keep track of the values of n for which Molasses(n) has already been called.)

[3 marks]

Use a dictionary, with the values of n as the keys and the values of Molasses(n) as the data. Dictionary lookup is very fast.

Marking: if they say 'dictionary' with no explanation, 2/3. If they say anything other than 'dictionary', 1/3. If they leave it blank, 0/3

Write a Python function **uses_Molasses** that takes a list of integers as its parameter, and returns a list containing the results of calling Molasses(n) on each number in the list, including the duplicates. Your function must call Molasses(n) as few times as possible. **You may assume that the function Molasses(n) has already been defined for you.**

For example, suppose Molasses(300) returns 8 and Molasses(77) returns 5. Then the instruction

```
print uses_Molasses([300,300,77,300,77])
```

should produce output

```
[8,8,5,8,5]
```

Write your function in the box on the next page:

[7 marks]

```
def uses_Molasses(my_list):
     md = {}
     result = []
     for n in my_list:
          if n not in md:
               md[n] = Molasses(n)
          result.append(md[n])
     return result
```

Marking: at least 4/7 if they have the right idea. If they are not using a dictionary, their solution should make correct use of whatever they chose to use. 0/7 is reserved for blank pages.

Question 4 (10 Marks):

Design a Python class called CD_Wallet. Each CD_Wallet object should be created with two attributes:

- **capacity**, which represents the number of discs the wallet can hold
  - the default capacity is 12
- **contents**, which is list of discs in the wallet (each disc will be represented by a string containing the title)
  - the default content list is an empty list

Your class must include an **__init__** function.

Your class must include an instance method called **add_CD(title)** which adds another disc to the wallet if there is room for it. If the wallet is already full, **add_CD(title)** should print a message saying "Wallet is already full"

Here is an example of some code that uses your class definition.

```
my_wallet = CD_Wallet(capacity=2, contents=["Polka Hits","Howling Dogs"])
my_wallet.add_CD("Grinding Gears")
```

Write your class definition in the box on the next page.

Answer Question 4 here:

```
class CD_Wallet(object):

     def __init__ (self,
                   capacity = 12,
                   contents = []
                   ):
          self.capacity = capacity
          self.contents = contents[:]

     def add_CD(self,title):
          if len(self.contents) < self.capacity:
               self.contents.append(title)
          else:
               print "Wallet is already full"
```

Marking: same rubric as before: 50% for having the right idea, even if there are many small errors. In the __init__ function, they need to get the 'self' usage correct when assigning values to attributes – max 8/10 if they get this wrong. Light penalization for getting the "self" stuff screwed up elsewhere.

As always, 0/10 is reserved for students who do not attempt the question.

Question 5 (10 Marks):

What "big O" order class would be most appropriate to describe the running time of each of the following Python functions. You may assume that easygui has been imported:

```
def Q5_1():
     n = integerbox("Enter an integer")
     x = n*n
     for i in range(x):
          print i,x
     for j in range(n):
          print j
```

Write your answer in this box

[5 marks]

O(n^2) - the i loop is the dominant part

Marking: they don't need to give an explanation. Give part marks as you feel appropriate.

```
def Q5_2():
     n = integerbox("Enter an integer")
     x = n*n
     for i in range(x):
          print i,x
          for j in range(n):
               print i,j,x
```

Write your answer in this box

[5 marks]

O(n^3) – the O(n) loop is inside the O(n^2) loop

Marking: as above

BONUS CANADIAN TRIVIA QUESTION (0 MARKS):

David Vernor of Ottawa was known throughout the latter part of his life as “The Professor” and was considered to be the world's greatest expert in his field. What was his field?

The rest of this page can be used for extra answer space or for rough work.