CS 115 Exam 4, Fall 2012

Your name: ____

Rules

- You may use one handwritten 8.5 x 11" cheat sheet (front and back). This is the only resource you may consult during this exam.
- Explain/show work if you want to receive partial credit for wrong answers.
- As long as your code is correct and obeys the specifications, you will get full credit. No points for style.
- When you write code, be sure that the indentation level of each statement is clear.

Grade (instructor use only)

	Your Score	**Max Score**
Problem 1		15
Problem 2		10
Problem 3		10
Problem 4		30
Problem 5		35
Total		100

Consider the following function definition:

```
def mystery_function(x):
    if x <= 0: return 0
    return x + mystery_function(x - 1)
```

A. What does the following function call return?
```
mystery_function(0)
```

B. What does the following function call return?
```
mystery_function(1)
```

C. What does the following function call return? (Show your work if you're worried about your arithmetic):
```
mystery_function(6)
```

D. How would you summarize what this function does in just a few words?

   Don't explain the code line-by-line. Provide a higher-level description like "computes *x* factorial" or "reverses the string *x*".

Problem 2: Selection sort (10 points)

Consider the following list:

```
L = ['CA',
     'TX',
     'NY',
     'FL',
     'IL',
     'PA',
     'OH',
     'MI' ]
```

and the following selection sort code (which works identically to your lab code, but is shorter and less efficient):

```
def selection_sort(list_to_sort):
  for i in range(len(list_to_sort) - 1):
    min_index = find_min_index(list_to_sort[i:]) + i
    list_to_sort[i], list_to_sort[min_index] =
      list_to_sort[min_index], list_to_sort[i]

def find_min_index(L):
  return L.index(min(L))
```

In the diagrams below, show the contents of the list after each of the first 4 iterations of the for-loop in selection_sort. If the list does not change from one iteration to the next, you can write "SAME" for the next iteration.

INDEX	**INITIAL ORDER**	**AFTER i=0 ITERATION**	**AFTER i=1**	**AFTER i=2**	**AFTER i=3**
0	CA				
1	TX				
2	NY				
3	FL				
4	IL				
5	PA				
6	OH				
7	MI				

Problem 3: Object-oriented programming terminology (10 points)

Consider the following Python code:

```
class Q:
   def __init__(self, x, y):
       self.x = x * 2
       self.y = y / 2

   def __str__(self):
       return 'Hi!'

   def q(self):
      return x * y

a = 5
c = Q(a, 4)
d = Q(10, 12)
print(a)
print(c)
```

A. What is the data type of the variable `a`?

B. What is the data type of the variable `c`?

C. What is the data type of the variable `d`?

D. List the methods of class `C`.

E. Label all of the calls to these methods with the names of the methods they call.

Problem 4: Creating classes (30 points)

In this problem, you will define a class to represent … a class! That is, you will define a class to represent a course at SSU.

If you use the input() or print() functions in your solution to this problem, you're doing it wrong!

Your class should be named `Course`, and you should define the following methods:

`__init__`: This method initializes a `Course` object.

- Parameters: department (e.g. 'CS'), course number (e.g. 115), and number of units (e.g. 4)
- Should initialize internal variables for department, course number, and number of units based on the information that was passed in. You can assume that the course number is actually numeric (don't worry about numbers like '115W').

`__str__`: This method returns a string that contains the object's information in the following format:

```
CS 115 (4 units)
```

`__lt__`: This method compares self to another `Course` object. It returns `True` if `self`'s course number is lower than the other object's course number and `False` otherwise. For example, if `self` is CS 115 and the other course is CS 215, it should return `True`.

`get_dept`: This method returns the object's department (e.g. 'CS').

`get_course_number`: This method returns the object's course number (e.g. 115)

`get_units`: This method returns the object's number of units (e.g. 4)

`is_lower_division`: This method returns `True` if the object's course number is less than 300 and `False` otherwise.

[WRITE YOUR PROBLEM 4 CODE HERE]

Problem 5: Using classes (35 points)

For this problem, you must write a **complete program**. However, you can assume that the `Course` class from Problem 4 has already been correctly defined for you.

To earn full credit, you must use the methods of the `Course` class whenever possible.

Read the instructions carefully before you start coding!

Your program should contain the following:

1. A function called `CreateCourse` to do the following:
   - Ask the user to enter the department, course number, and units all on the same line. For example:
     ```
     CS 115 4
     ```
   - If the user's line is blank, ***return*** `None`.
   - Otherwise, if the user did not provide exactly 3 values separated by whitespace, exit the program with an error message.
   - If the course number and/or the number of units are not POSITIVE numbers, exit the program with an error message.
   - Otherwise, create ***and return*** a `Course` object that uses the information the user entered.

2. A `main` function to do the following:
   - Call `CreateCourse` repeatedly until the user enters a blank line instead of the course information.
   - Then, ask the user to input the name of one department. You may assume that there is at least one course in that department.
   - Using your `Course` methods, print each course in that department (and only that department) using the format
     ```
     CS 115 (4 units)
     ```
   - Print the ***total*** number of units in that department's courses.
   - If the department has no lower-division courses, print
     ```
     No lower-division courses.
     ```
   - If the department has at least one lower-division course, print the course with the lowest number, as follows:
     ```
     First course:
     CS 101 (3 units)
     ```

[WRITE YOUR PROBLEM 5 CODE HERE]

[EXTRA SPACE]