CS 115 Exam 4, Spring 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, you will get full credit. No points for style.
- When you write code, be sure that you clearly indicate the indentation level of each statement.

Grade (instructor use only)

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

Problem 1: Object-oriented programming terminology (15 points)

Consider the following Python program:

```
class Y:
   def __init__(self, a1):
       if a1 > 0: self.b = a1
       else: self.b = 0

   def f1(self):
       self.b *= 2

   def f2(self, c):
      if self.b > c: return self.b
      return c

y1 = Y(5)
```

Write in one of the following responses to each of the statements below:

- class
- object
- method
- instance variable
- function parameter

a) `Y` is a(n)…

b) `f1` is a(n)…

c) `a1` is a(n)…

d) `y1` is a(n)…

Problem 2: Selection sort (10 points)

Consider the following unsorted list:

```
dwarfs = ['Snow White',
          'Doc',
          'Grumpy',
          'Bashful',
          'Sleepy',
          'Happy',
          'Sneezy',
          'Dopey']
```

and the following selection sort (which is essentially identical to your lab code):

```
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	Snow White				
1	Doc				
2	Grumpy				
3	Bashful				
4	Sleepy				
5	Happy				
6	Sneezy				
7	Dopey				

Problem 3: Dictionaries (15 points)

This question asks you to perform various operations on a Python dictionary called *standings* that maps the names of baseball teams to the number of games they have won this season.

(a) Write a line of code to create a dictionary named *standings* and give it the following initial mappings:
- The Dodgers have won 19 games.
- The Giants have won 15 games.
- The Padres have won 11 games.

(b) Write a line of code to add the following mapping to the *standings* dictionary:
- The As have won 16 games.

(c) Write code to do the following. Do NOT assume anything about the current contents of the *standings* dictionary:
- If the Angels are in the dictionary, increase their number of wins by 1.
- If the Angels are not in the dictionary, add them to the dictionary with a total of 1 game.

Problem 4: Implementing classes (30 points)

In this problem, you will define a class to represent an SSU student.

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

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

`__init__`: This method initializes a Student object.

- Parameters: a name, a GPA, and a number of units taken
- Should initialize instance variables for name, GPA, and units based on the information that was passed in. If the GPA or number of units is negative, sets it to 0. (Don't worry about non-numeric values in this method.)

`update`: This method updates the instance variables of the Student object if the Student takes a new class.

- Parameters: units for the new class, grade points earned (as a number) in the new class.
- Should modify the instance variable for units to add the units for the new class
- Should modify the GPA to incorporate the grade earned in the new class. (Note that this will be a weighted average using both the unit counts and both sets of GPAs.)

`get_gpa`: This method should return the student's GPA.

`get_name`: This method should return the student's name.

Problem 5: Using classes (30 points)

For this problem, you must write a **complete program**. That includes a docstring, a def main(), any necessary library imports, etc.

You can assume that the Student class from Problem 4 has already been correctly defined.

Read the instructions carefully before you start coding!

Your program should contain the following:

1. A function called CreateStudent to do the following:
    - Ask the user to enter a string, a floating-point number, and an integer (student's name, GPA, and units)
    - Print an error message and exit the program if the user did not enter numeric values for the GPA or the units.
    - Otherwise, create a Student object with those values.
    - Return the newly created Student

2. A main function that does the following:
    - Call Student twice to create two Student object.
    - For each of the two students, repeatedly asks the user to enter the number of units and grade point for the courses the student has taken and updates the Student object each time. It should stop asking about a student when the user enters zero units for that student.
    - Using your methods, prints the name of the student with the higher GPA. If the students are tied, you should print both names..

Bonus: Recursion (+5 points)

Consider the following function definition:

```
def f(x):
     if x == 0: return 0
     return 1 + f(x // 10)
```

---

a) What does the following function call return?

```
f(25480)
```

b) Suggest a more helpful name for this function that succinctly describes its purpose: