Quantitative Reasoning 20
Final Exam, May 14, 2010

Work all parts of all 4 problems on front and back of the exam. The points for each question are the weight of the question in the total exam grade and are not an indication of how long it might take you to complete the problem. They total 100 points. DO NOT WORK MORE THAN 40 MINUTES ON ANY PROBLEM UNTIL YOU HAVE TRIED THEM ALL. The programs you are asked to write should be short. If you are asked to write a function, you may always break it down to several sub functions if you like. Suggested answers will be posted on the course web site tomorrow. We can not give out grades over the phone or email (or when passing on the street). The grades will be posted on the Registrar's information system late next week.

1a.(10 points) Write a Python function which accepts an unsorted list of numbers as an argument, and returns the largest and smallest values in the list AND the list with the extremes removed. For example, if your function is named extremes, then

```python
a=[4,2,3,6,1,3]
high,low,a=extremes(a)
```

would set high=6, low=1 and leave a with the value [4,2,3,3]. You may assume that there are at least 2 items in the list. If there are ties for the highest and/or lowest values, you should just remove 1 of each.

1b.(10 points) The function in part 3a can be easily used in a recursive sort. Write it as a recursive Python function.

1c.(5 points) What is the efficiency of your recursive sort from part 3b? Specifically, about how many if commands (in the extremes function) are executed when using it to sort a list of n items? How does this efficiency compare to other sorts we have studied in the course?

2.(30 points) Suppose you are hired to write a Python GUI to calculate the average score for each competitor in a figure skating contest. There are 7 judges who will enter their scores in the GUI. When an “average” button is pushed, the seven scores will be read, the highest and lowest scores will be thrown away and the skater given the average of the 5 remaining scores. That average will then be displayed in a label next to the “average” button. Finally, there should be a “clear” button that clears the entries and score display for the next skater. Show all commands needed for the program to work, EXCEPT that you may call your function from part 1a, above, without rewriting it here. The GUI might look something like this (the appearance of the individual widgets, border, background, etc., might be different):

```
Judge 1  Judge 2  Judge 3  Judge 4  Judge 5  Judge 6  Judge 7
8.4     8.1     8.7     9.3     8.3     8.3     8.8

average 8.5

clear
```

Comment everything so that the grader has no problem in understanding what your program and its functions should do.

OVER, more on the back…
3. (15 points) The following Python function (turned in without comments, Tsk,Tsk,Tsk) is supposed to remove redundant items from a list. For example, purge([1,2,1,3,4,2,1,5,1]) should return [1,2,3,4,5]. Unfortunately, it doesn’t work. It may have syntactic errors, runtime errors and/or logical errors. Find all of them and suggest a correction for each.

```python
def purge(data):
    i=0
    while i<length(data):
        for item in data[:i]:
            if item=data[i] then
                data=data[:i-1]+data[i+1:]
                break
        i=i+1
    return data
```

4. A graph is a list of points, sometimes called nodes, and a list of pairs of indeces of points in the list, each pair indicating that a line (sometimes called an edge) should be drawn from one of the points to another, a connection between them. All parts of this problem are straightforward. Don’t make them harder than they are.

4a. (5 points) Define a class, named node, such that each instance of the class has an attribute named x and one named y, the coordinates of the point. The constructor function for the class should take two arguments, the x and y values.

4b. (5 points) Define a class, named graph, such that each instance of the class has two attributes, the first a list of instance objects of the class, node, and the second a list of connections. The 2 lists should be arguments to the constructor for the class.

4c. (10 points) Write a Python function which creates a random graph. Specifically, It should generate a list of instance objects of the class node whose length is a random integer from 3 to 10 and with each of them having their x and y values a random integer from 0 to 400. It should then generate a list of ordered pairs, with the length of the list a random integer between the number of nodes and 5 times the number of nodes. The start and stop nodes of each pair should be different, but don’t worry about redundant connections in your list. The function should then create an instance object of class graph with the 2 random lists as attributes and return this instance object. Include any necessary imports in your function.

4d. (10 points) Write a Python function which accepts an instance object of the class, graph, and displays it on a Tk canvas. Your function should assume that all nodes will be on a 400 by 400 canvas (eg, such as for the random graph from 4c). A line should be drawn on the canvas for every connection in the list of connections for the graph instance object. Again, include any necessary imports or other commands in your function needed to make it work.