Name: ___**Sample Solution**_____

Email address: _____

Quiz Section: _____

CSE 140 Winter 2014: Final Exam

(closed book, closed notes, no calculators)

Instructions: This exam is closed book, closed notes. You have 50 minutes to complete it. It contains 11 questions and 10 pages (including this one), totaling 90 points. Before you start, please check your copy to make sure it is complete. Turn in all pages, together, when you are finished. Please write neatly; we cannot give credit for what we cannot read.

Good Luck!

Total: 90 points. Time: 50 minutes.

Page	Max Points	Score
2	**9**	
3	**12**	
4	**12**	
5	**16**	
6	**6**	
7	**10**	
8	**9**	
9	**6**	
10	**10**	
Total	**90**	

1) [3 pts] What error would the following code produce? If there is more than one error, give the FIRST one that would be reported.

```
lst = [0, 15, 22, 11, 4]
x = lst[5]
y = x+x
print "y"
```

x = lst[5]

IndexError: list index out of range

2)[3 pts] Write code that would produce a "`TypeError: 'list' object not callable`" error

```
lst = [1,2,3]
lst(4)
```

3) [3 pts] In theory what distribution corresponds to a histogram where each bin has the same height? (i.e. - the ideal histogram for rolling a die)

Uniform distribution

4) [12 pts] Write at least two assert statements for each function below that you might use to test that the following functions are correctly implemented. If you think that the functions are incorrectly implemented, include assertions that will expose errors in the function.

```
def max(lst):
    """ Takes a list of integers and returns the largest value
    in the list. We consider the maximum value of an empty list
    to be None. """
    max = 0
    for x in lst:
        if x > max:
            max = x
    return max
```

assert max([-3, -5, -7]) == -3

assert max([]) == None

```
def avg(lst):
    """ Takes a list of integers and returns a float which is the
    average of the list. We consider the average value of an
    empty list to be None. """
    total = 0
    num_values = 0
    for x in lst:
        total += x
        num_values += 1
    return total / num_values
```

assert avg([]) == None

assert avg([7, 8]) == 7.5

5) [12 pts] Implement the following functions:

```
def normalize(lst):
    ''' Returns a new list in which the elements of lst are all
    scaled such that the elements of the new list sum to 1. The
    input is a list of non-negative numbers whose sum is > 0.'''


    list_sum = float(sum(lst))

    result = []

    for elt in lst:

        result.append(elt / list_sum)

    return result


assert normalize([]) == []
assert normalize([1]) == [1.]
assert normalize([1, 1]) == [0.5, 0.5]
assert normalize([1, 4]) == [0.2, 0.8]

def rolls_to_hist(rolls):
    ''' Convert a list of rolls of a single 6-sided die to a list
    of counts. rolls is a list of numbers whose values are
    between 1 and 6 inclusive. In the result list, result[i] is
    the number of times i appeared in the input list. '''


    result = [0,0,0,0,0,0,0]

    for roll in rolls:

        result[roll] += 1

    return result




assert rolls_to_hist([]) == [0, 0, 0, 0, 0, 0, 0]
assert rolls_to_hist([1]) == [0, 1, 0, 0, 0, 0, 0]
assert rolls_to_hist([1, 2, 3, 4, 5, 6]) == [0, 1, 1, 1, 1, 1, 1]
assert rolls_to_hist([1, 1, 2, 2]) == [0, 2, 2, 0, 0, 0, 0]
```

6) [16 pts] The following functions are in a single file. Implement the function bodies. We are interested not just that the code works, but that you have implemented the functions in good style. Imagine this is code submitted for a homework assignment.

```
import random

def single_die_roll(k):
    """Return a single roll of a k-sided die."""


    return random.randint(1, k)

    # OR return random.randrange(k) + 1


def dice_roll(numdice, k):
    """Return the sum of the given number of rolls of a k-sided
    die."""


    result = 0

    for i in range(numdice):

        result = result + single_die_roll(k)

    return result


def dice_roll_trials(numdice, k, numtrials):
    """Returns a list of numtrials numbers, each the sum of
    rolling numdice k-sided dice."""


    result = []

    for i in range(numtrials):

        result.append(dice_roll(numdice, k))

    return result
```

b) Write code to print a list of 100 trials, of 3 rolls of a 8-sided die. In other words, print a list of 100 trials, of rolling 3 8-sided die.

MY ANSWER:

```
print dice_roll_trials(3, 8, 100)
```

7) [6 pts] Write a docstring for the following function. Document the inputs and any outputs or side effects.

```
import matplotlib.pyplot as plt


def plot_func(func, x_values):
    """    Your docstring will go here    """
    y_values = []


    for x in x_values:
        y_values.append(func(x))


    plt.clf()
    plt.plot(x_values, y_values)
    plt.show()
```

MY ANSWER:

*** Inputs:**

func is a function that takes a single argument, and returns a number

x_values is a list of numbers

*** (No outputs)**

*** Side effects:**

Clears any previous matplotlib graph

Displays a graph of func(x) vs x for all x in x_values.

8) [10 pts] You are given the following code:

```
def a(b):
    f = 0
    g = 0
    c = open(b)
    for d in c:
        e = int(d.strip())
        f = f + e
        g = g + 1
    c.close()
    return float(f) / g
```

a) This function has terrible variable naming. Next to each letter below, write a better name for that variable:

a **__average_from_file** _____(the name of the function)

b **___filename** _____

c **___input** _____

d **___line** _____

e **___number** _____

f **___total** _____

g **___num_values**_____

b) Below, write a good docstring for this function.

“””Given filename, the path name of a file that contains one integer on every line,

returns the average of the integers in that file.”””

9) [9 pts] You are given the following class definition:

```
class RatNum:
    """Represents a rational number such as 1/2 or 22/7."""

    def __init__(self, numerator, denominator):
        """Construct a RatNum with the given numerator and
        denominator."""

    def add(self, other):
        """Return a new RatNum that is the sum of this RatNum and
        RatNum other."""

    def to_float(self):
        """Return a float that approximately equals the value of
        this RatNum."""
```

a) Write code that creates a variable r1 bound to a RatNum object representing 4/5 and a variable r2 bound to a RatNum object representing 6/78

```
r1 = RatNum( 4, 5 )

r2 = RatNum( 6, 78 )
```

b) Write code that prints the float value of the sum of r1 and r2

```
print r1.add(r2).to_float() # OR

total = r1.add(r2)
print total.to_float() # OR

total = RatNum.add(r1, r2)
print total.to_float() #OR print RatNum.to_float(total)
```

c) List one advantage of using a class

Many possible answers. Allows the implementer of the RatNum class to change the internal implementation while the client code can remain unchanged. Collects the data representation and functions operating on that data in one place.

10) [6 pts] Write the base case(s) for the following function:

```
def sum_list(lst):
    ''' Returns the sum of the elements in lst
    Assumes that lst is a list of numbers. '''
    # PUT YOUR CODE HERE:

   if lst == []:    # or if len(lst) == 0:
        return 0

    return lst[0] + sum_list(lst[1:])
```

11) [10 pts] a) **Draw** the entire environment, including all active environment frames and all user-defined variables, **the first time return 0 is executed**. Feel free to draw out the entire environment, but be sure to CLEARLY indicate what will exist the first time return 0 is executed.

b) What does this function calculate?

MY ANSWER:

Multiplies a times b

```
def mystery(a, b):
    if b == 0:
        return 0
    elif b < 0:
        return -1 * mystery (a, -b)
    else:
        return a + mystery (a, b - 1)

print mystery (4, -2)
```