Quiz 1
6.00 Spring 2013

Name:	
Athena user name:	
Recitation Section:	

Question 1. (25 pts)

For each of the following blocks of code, write the output that will be printed by the code.

1.a)

```
letters = ['a', 'b', 'c', 'd']
word = ''
for x in letters:
   word  = word + x + 'x'
print word
```

Answer:

1.b)

```
x = 'Global Variable'
def output(x):
      print x

output('local value')
```

Answer:

1.c)

```
x = 'A global variable'
def moo(x):
   return x + ', for sure'
def foo(x):
   x  = 'the wrong thing'
   return x
def output(x):
   x = 'or a parameter'
   x = foo('a good choice')
   return moo(x)

print output('What will I print?')
```

Answer:

1.d)

```
def f(N):
    x = 1.0/N
    sum = 0.0
    for i in xrange(N):
        sum = sum + x;
    if sum == 1.0:
        print 'YES'
    else:
        print 'NO'
f(10)
```

Answer:

1.e)

```
x = 33
y = x / 2
t = 20
t = t + 13.0
v = t / 2
print str(y) + ' and ' + str(v)
```

Question 2. (15 pts)

Provide short answers to the following questions

2.a) What number in decimal notation do the following binary mantissa and exponent represent: [101, -10] ?

Answer:

2.b) How do we call the following two types of knowledge (one word each):
1/ The square root of x is the number y such that $y^2=x$
2/ The Newton-Raphson algorithm can compute a square root through successive approximations. At each step, the current solution is refined by solving a linearized equation based on the derivative at the current guess.

Answer:
1/
2/

2.c) What does the code below compute?

```
N=1000
NC=0.0
x=-1.0
for i in xrange(2*N):
    y=-1.0
    for j in xrange(2*N):
        if x*x+y*y<1:
            NC=NC+1
        y=y+1.0/N
    x=x+1.0/N
print NC/(N*N)
```

Answer:

Question 3. (10 pts)

You are given a list called words with 2000 words sorted in alphabetical order. Below are three algorithms that you may use to find whether a target word is in the list.

```
def find1(words, target):
    for x in words:
        if x == target:
            return True
    return False
```

```
def find2(words, target):
    size = len(words)
    pos = size/2
    if(words[pos] > target):
        return find2(words[pos:size], target)
    else:
        return find2(words[0:pos], target)
```

```
def find3(words, target):
    low = 0;
    high = len(words)
    while( low < high):
        pos = (low + high)/2
        if (words[pos] > target):
                high = pos
        if( words[pos]< target):
                low = pos + 1
        if( words[pos] ==target):
                return True
    return False
```

3.a) One of the algorithms above is asymptotically slower, one of them is faster, and one of them is just wrong, can you tell which one is which and why (in one sentence)?

Slower:
Why:

Faster:
Why:

Incorrect:
Why:

Question 4. (20 pts)

Given a polynomial represented as a list of coefficients similar to pset 2 (where the index is equal to the order, from constant to higher exponents), write a function that evaluates the polynomial for a given input number. You can assume that the list is not empty.

```
def eval_poly(poly, x):
      """poly contains a list of floating point coefficients
      and x is a floating point number.
      Returns the value of the polynomial at x"""
```

Question 5. (20 pts)

In python there are many ways of reversing a given string **s**; for example, you can evaluate s[::-1]. For this problem, you goal will be to write your own reverse function to satisfy the following requirements:

- Your function must be recursive
- You function can not use any built-in python mechanism that would automatically reverse the string for you (you can't write s[::-1], for example).
- Some of the features that you can use and you will find very useful are string concatenation (a + b) and string range (s[a:b]).

```
def reverse(myString):
    """reversal function assumes myString is a string"""
```

Question 6 (10 pts)

Use the code below to answer the questions that follow. Note that function **f** assumes **L** has at least two elements.

```
def f(L, x):
    """The code assumes at least 2 elements in the list L"""
     def helper(L, x, low, high):
         if high-low<2:
             return 0
         m=(low+high)/2
         print m # note the print
         if L[m]>x:
             return helper(L, x, low, m)
         else:
             return helper(L, x, m, high)
     return helper(L, x, 0, len(L))
L=[1, 4, 5, 6, 13]
y=f(L, 4)
print 'final value:', y
```

6.a) What does this code print (note the print statement inside the function) ?

6.b) Is this algorithm iterative or recursive?

6.c) What is the name of the algorithm?

6.d) What is its complexity with respect to the length N of list L?

6.e) In addition to the size of the list. what key assumption does the algorithm make?

The following questions won't impact your grade but will help us improve the course.

How fast are the lectures? (circle one)

1. Too slow, I am bored
2. A little slow, I wish they went a little faster
3. Just right
4. A little fast, I wish they slowed down a little
5. Way too fast, If it wasn't for EdX/Office Hours/Recitations I'd be totally lost

How hard are the problem sets?

1. Too easy, I am bored
2. A little easy, I wish they were more interesting
3. Just right
4. A little hard, I wish they didn't take 10 hours to do
5. Way too hard

For each of the following, give a number from 0 to 5 based on how useful you have found it (0 is have never used it, and 5 is I couldn't live through this course without it).

1. Text Book _____
2. Lecture Slides _____
3. EdX site _____
4. Office Hours _____
5. Google _____