Name:	
Athena user name:	
Recitation Section:	

Question 1. (10 pts)	……._____
Question 2. (15 pts)	……._____
Question 3. (20 pts)	……._____
Question 4. (25 pts)	……._____
Question 5. (25 pts)	……._____
Question 6 (5 pts)	……._____

Question 1. (15 pts)

Identify the complexity of each of the codes shown below

a)

```
def sort1(L):
  doMore = True
  while(doMore):
    doMore = False
    for i in xrange(0, len(L)-1):
       if(L[i] > L[i+1]):
          swap(L, i, i+1)
          doMore = True
  return L
```

b, c)

```
def sort2(L):
  if(len(L) <= 1):
    return L
  mid = len(L) / 2
  L1 = sort2(L[0:mid])
  L2 = sort2(L[mid:])
  return combine(L1, L2)

def combine(L1, L2):
  result = []
  p1 = 0
  p2 = 0
  for i in xrange(0, len(L1) + len(L2)):
    if(decide(L1, L2, p1, p2)):
      result.append(L1[p1])
      p1 += 1
    else:
      result.append(L2[p2])
      p2 += 1
  return result

def decide(L1, L3, p1, p2):
  '''decide whether to read from L1 or
L2'''
```

a) Complexity of sort1:

b) Complexity of combine:

c) Complexity of sort2:

Question 2. (15 pts)

Object-oriented programming. We want to implement classes to store information about different items in a store. We give you a class Good that is used to store the name and price of regular item. Your task is to complete a derived class GoodOnSale that stores, in addition to the regular price already stored by Good, a percentage off.

```
class Good:
    def __init__(price, name):
        self.price=price
        self.name=name
    def getCurrentPrice():
        return price
```

```
class GoodOnSale(Good):
    def __init__(price, name, percentageRebate):
         #Your code here
```

```
    def getCurrentPrice():
        #Your code here
```

Question 3. (20 pts)

More object-oriented programming. We give you the following class for complex numbers (which is different from the one in lecture).

```
class ComplexNumber:
    def __init__(re, im):
        self.re=re
        self.im=im
    def mutateAdd(cplx2):
        self.re=self.re+cplx2.re
        self.im=self.im+cplx2.im
    def __str__():
        return str(self.re) + '+' + str(self.im) + 'i'
```

a) What does the following code print?

```
c1=ComplexNumber(0.0, 1.0)
c2=ComplexNumber(1.0, 0.0)
c3=ComplexNumber(0.0, 1.0)
c4=c1
c1.add(c2)

print c1
print c3
print c4
```

Answer:

b) What is wrong with the following code?

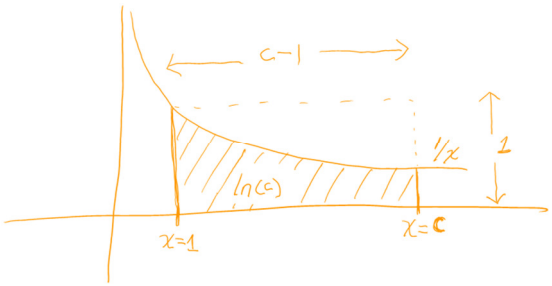
```
c1=ComplexNumber(0.0, 1.0)
c1.re=1.0
```

Answer:

Question 4. (25 pts)

For this exercise, we are going to use the Monte Carlo method to compute the natural logarithm of a value c > 1; i.e. ln(c). We know from basic calculus that the natural logarithm ln(c) equals the area under the curve 1/x between 1 and c (if you did not know that, you know it now!). We are going to compute the area under this curve by taking a random sample of the points in the rectangle between (1,0) and (c,1)---which has a known area (c-1)---and computing the fraction that fall below the 1/x curve as shown in the figure below.

Your code should raise a ValueError if C is not greater than 1.




```
def monteCarloLn(c, N):
   '''return a monte carlo approximation of ln(c) using N samples.
      raises a ValueError if c<=1 '''
```

Question 5. (25 pts)

In the game of Jenga, players take turn moving wooden blocks in an assembly until one of the players makes a move that causes the whole structure to collapse, in which case he or she loses. Write a Monte Carlo simulation that computes the average duration of a game of two players given the probability p1 and p2 that each of them loses at a given step. You should assume that the probabilities of losing are independent and remain constant as the game progresses.

```
def monteCarloJenga(p1, p2, N):
    '''run N trials of simulation to estimate the average duration of the game
       assuming that on every turn, player 1 has a probability p1 of losing and
       player 2 has a probability p2 of losing.'''
```

Question 6. (5 pts)

Link each of the concepts on the left with a concept on the right.

	abstraction barrier
memoryless property	
	Gaussian Distribution
overriding parent class methods	
	Exponential Distribution
getter and setter methods	
	polymorphism
package data and computation	
	Objects