This quiz is open book and open notes, but do not use a computer.

Please write your name on the top of each page, and your user name and the hour of the recitation you attend on the first page. Answer all questions in the boxes provided.

1) Are each of the following True or False (15 points)

- **F**  1.1. Let D1 and D2 be two different normal distributions. If a sample, s1, is one standard deviation from the mean of D1 and a sample, s2, is two standard deviations from the mean of D2, then s2 is always further (in an absolute sense) from the mean of D2 than s1 is from the mean of D1.

- **F**  1.2. In Python, a function cannot be used as a class variable.

- **T**  1.3. Increasing the number of buckets in a hash table typically decreases the number of collisions.

- **T**  1.4. For a Gaussian distribution decreasing the confidence interval (e.g., from ±5 to ±4) decreases the confidence level (e.g., from 95% to 90%).

- **F**  1.5. In Python, none of the standard methods of class object can be overridden by subclasses of object.
2) Consider the following code:

```python
def oneTest():
    tries = 0
    while True:
        tries += 1
        ind1 = random.choice(range(52))
        ind2 = random.choice(range(52))
        if ind1 == ind2: break
    return tries

def makePlots(numTrials, oneTest):
    numTries = []
    for t in range(numTrials):
        numTries.append(oneTest())
    pylab.plot(numTries)
    pylab.figure()
    pylab.hist(numTries, bins = 10)
makePlots(1000, oneTest)
```

2.1. Write Python code that calculates the width of each bar (they are all the same width) in the histogram. (5 points)

```python
minVal, maxVal = pylab.xlim()
width = (maxVal - minVal) / 10.0
```

2.2. Assuming that the width of each bar is \( w \), describe the range of values on the x axis covered by the tallest bar in the histogram. (5 points)

```python
(minVal, w + minVal)
```

2.3. Is it likely that the call to `pylab.plot` would produce a plot similar to one below? (5 points)

```
Y
```
3) John had a strategy for eventually winning a lottery with 1000 tickets. The first time he entered he would buy one ticket. If he didn’t win he would double the number of tickets (to two) he bought the next time he entered. If he didn’t win that time, he would double the number of tickets (to four) again. Etc. What is the probability that John wins the lottery \textbf{before} playing it 4 times? (10 points)

The chance of John winning the first time is \(\frac{1}{1000}\) (assuming the tickets are uniformly distributed).

If he doesn’t win, he enters the second lottery. The probability of his not winning the first is \(\frac{999}{1000}\). The probability of his winning the second lottery is \(\frac{2}{1000}\).

Applying the same argument for the third lottery we have the probability of John winning the lottery before playing it 4 times is:
\[
\frac{1}{1000} + \frac{999}{1000} \times \frac{2}{1000} + \frac{999}{1000} \times \frac{998}{1000} \times \frac{4}{1000}
\]

Alternately, one can compute the probability of John not winning the first lottery AND not winning the second AND not winning the third. These are independent events and we can write this probability as: \(\frac{999}{1000} \times \frac{998}{1000} \times \frac{996}{1000}\).

The probability of John winning before playing the lottery 4 times is:
\[
1 - \left(\frac{999}{1000} \times \frac{998}{1000} \times \frac{996}{1000}\right)
\]
which is the same as the previous answer.
4) Write a function that uses a Monte Carlo simulation to estimate the probability of John winning the lottery within \( n \) attempts, assuming he uses the strategy of problem 3. Assume that 10,000 trials are sufficient to provide an accurate answer. You may call the function:

```python
def runLottery(ticketsSold, ticketsBought):
    """ticketsSold is the number of tickets sold in a lottery and ticketsBought is the number of tickets bought by John. It returns 1 if John won the lottery and 0 otherwise.""
```

(20 points)

```python
def sim(n, ticketsSold):
    #write your code below
    numWins = 0
    for i in range(10000):
        for j in range(n):
            if runLottery(ticketsSold, 2**j):
                numWins += 1
                break
    return float(numWins) / 10000
```
5) What does the following code print? (20 points)

```python
class Shape(object):
    def __lt__(self, s1, s2):
        return s1.area() < s2.area()
    def __str__(self):
        try:
            return 'Shape with area ' + str(self.area())
        except:
            return 'Shape'
class Rectangle(Shape):
    def __init__(self, h, w):
        self.height = h
        self.width = w
    def __lt__(r1, r2):
        return r1.height < r2.height
class Square(Rectangle):
    def __init__(self, s):
        Rectangle.__init__(self, s, s)
    def __str__(self):
        return 'Square with side ' + str(self.height)
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return 3*(self.radius**2)
def reorder(L):
    for e in L:
        if L[0] < e: L[0] = e

s = Square(5)
s1 = s
s = Square(4)
r = Rectangle(3, 4)
c = Circle(0.5)
L = [s, s1, r, c]
try:
    reorder(L)
except:
    print 'here'
for e in L: print e

here
Square with side 5
Square with side 5
Shape
Shape with area 0.75
```
6. An experiment was run that involved collecting one data value every second for 20 seconds. When plotted the data looked like,

![Graph of data](image)

Write code that finds an appropriate model for this data, and then uses that model to plot projected values for the next 80 time units. Assume that the 20 data points are in an array named `vals`. (10 points)

```python
import pylab

a, b, c = pylab.polyfit(pylab.arange(20), vals, 2)
new_x = pylab.arange(20, 100)
new_y = a * (new_x ** 2) + b * new_x + c
pylab.plot(new_x, new_y)
pylab.show()
```
7) Next to each item in the left column write the letter labeling the item in the right column that best matches the item in the left column. No item in the right column should be used more than once, and no box should contain more than one letter. (10 points)

| e | inheritance | a) O(n log n) |
| g | standard deviation | b) O(n**2) |
| f | half-life | c) O(n) |
| d | exception | d) try |
| a | merge sort | e) polymorphism |
|   |               | f) exponential |
|   |               | g) variance |

8) Do you think that the lectures are too slow paced, too fast paced, about right?

Too slow  1  2  3  4  5  Too fast

9) How long are you spending on the problem sets?

Hours per problem set:  <6  6-10  10-15  15-20  >20