Question 1. (5 pts)

Are each of the following True or False?

T	Agglomerative hierarchical clustering (Bottom to top) involves O(n) agglomerations.
F	The greedy solution to the Knapsack problem always finds the optimum.
T	K-means clustering computes new centroids locations at each iteration.
T	The greedy solutions to the knapsack problem is O(n log n).
F	Correlation always implies causation.

Question 6. (10 pts)

In a groundbreaking scientific study, researchers at the University of Candyland tested the correlation between jellybeans of 20 different colors and acne. They divide their human subjects into 20 groups of 100 people each, where each group eats only one color of jelly beans. One month later, they record the percentage of subjects who have acne in each group and they find that while the average is 31%, the group of subjects who ate **green** jelly beans has a much higher percentage with acne, 42%.

Below we've written part of a Monte-Carlo simulation to determine the probability that this occurred by chance. Your goal is to complete the code.

```
import random
PROB_ACNE = 0.31

def monteCarloTest(N):
      '''Compute the probability of getting the surprising result'''
      good = 0.0
      for x in xrange(0, N):
      if(surprisingResult()):
            good += 1.0
      return good / N

def surprisingResult():
    ''Simulate the experiment assuming that a person has a
      uniform probability PROB_ACNE of getting acne. Return true if
     the experiment supports the published result; i.e. if more than
     42 people in some group have acne. Otherwise return false'''
    #Your code here

    #check for each group
    for i in xrange(num_groups):
        acne_count=0
        #perform trial for all subjects in the group
        for j in xrange(subjects_per_group):
            if random.random()<=PROB_ACNE:
                acne_count+=1
        #check if at least 1 group is above 42%
        if float(acne_count)/subjects_per_group >.42:
            return True
    #no group is above 42%
    return False

print 'Probability of getting the surprising result = ' + str(monteCarloTest(1000))
```

Question 8. (5 pts)

```
import pylab
import random as rr
xVals=[]
yVals=[]
for i in xrange(1000):
      x=rr.random()
      y=x*x+2
      xVals.append(x)
      yVals.append(y)
xVals=pylab.array(xVals)
yVals=pylab.array(yVals)
a, b, c, d=pylab.polyfit(xVals, yVals, 3)
print round(a), round(b), round(c), round(d)
```

What does this code print?

```
-0.0 1.0 -0.0 2.0
```

Question 9. (5 pts)

Link each concept on the left to the one on the right that is most related.

coefficient of determination	proper scaling
feature vector	avoid overfitting
LIFO	half life
cross-validation	depth-first search
exponential distribution	goodness of fit