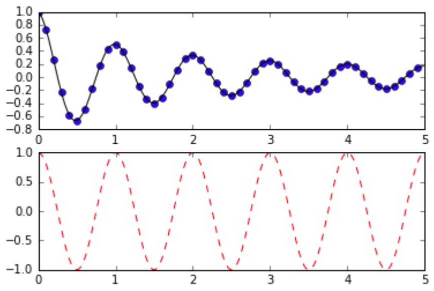
Answer Key: MAT 456-01 Final Exam, Spring 2016

1. What will the following code draw:

```
import numpy as np
import matplotlib.pyplot as plt

def f(t):
    return np.cos(2*np.pi*t)/(t+1)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

Answer Key:



2. For each of the regular expressions, give a string that will matches it:

   (a) `(\d\w){2}\d`

   (b) `[bch][ai]+[tT]+`

   (c) `((Roosevelt)|(Long)|(Ellis)) Island`

   (d) `\d{3}-\d{2}-\d{4}`

   (e) `\w+@\w+\.\w+`

Answer Key:

Note: there are multiple possible answers for each:

   (a) `1a2b3`

   (b) `baaaaT`

   (c) `Roosevelt Island`

(d) `123-45-6789`

(e) `herbert@lehman.cuny.edu`

3. The New York City Open Data project contains all motor vehicle collisions reported to the New York Police Department. The data can be downloaded as CSV files with the following format:

```
DATE,TIME,BOROUGH,ZIP CODE,LATITUDE,LONGITUDE,LOCATION,ON STREET NAME,CROSS STREET NAME,OFF STREET
02/01/2016,0:09,BRONX,10465,40.8341548,-73.8174815,"(40.8341548, -73.8174815)",BARKLEY AVENUE,DEAN
```

All lines are formatted similarly: they start with the date, then time, the borough, zip code, latitude and longitude, and also include cross streets, types of vehicles involved, number of injuries/fatalities, and possible cause. The first line of the file gives the entries in the order they occur in the rows.

Write a program that takes a file, `bronxCollisions.csv`, and prints out all the zip codes for crashes occur in the Bronx:

Answer Key:

```
import csv

#Using the dictionary reader to access by column names
f = open("bronxCollisions.csv")
reader = csv.DictReader(f)
m = [row[''ZIP CODE'] for row in reader if row['BOROUGH'] == 'BRONX']
f.close()
print m
```

Another way to do this, using the regular csv reader (ignoring first line with column names):

```
import csv

#Note the use of 'with' for files:
with open("bronxCollisions.csv") as f:
     reader = csv.reader(f)
     reader.next()  #Ignore line with column names
     m = [row[3] for row in reader if row[2] == 'BRONX']
print m
```

4. The Center for Disease Control (CDC) provides data on the number of occurrences of Lyme Disease. Assuming you have the data stored:

```
years = [2003,2004,2005,2006,2007,2008,2009,2010,2011]
ny = [5399,5100,5565,4460,4165,5741,4134,2385,3118]
nj = [2887,2698,3363,2432,3134,3214,4598,3320,3398]
ct = [1403,1348,1810,1788,3058,2738,2751,1964,2004]
```

Write a program that will plot the state numbers as well as the total Lyme Disease occurrence (i.e. the sum of the values for the three states for each year).

Answer Key:

```
import matplotlib.pyplot as plt

#Data from CDC's Lyme Disease page:
years = [2003,2004,2005,2006,2007,2008,2009,2010,2011]
ny = [5399,5100,5565,4460,4165,5741,4134,2385,3118]
nj = [2887,2698,3363,2432,3134,3214,4598,3320,3398]
ct = [1403,1348,1810,1788,3058,2738,2751,1964,2004]

totals = [i+j+k for i,j,k in zip(ny,nj,ct)]

#Create scatter plots for each state:
plt.scatter(years, ny, label="NY", c = "blue", s=75)
plt.scatter(years, nj, label="NY", c = "red", s=75)
plt.scatter(years, ct, label="NY", c = "purple", s=75)
plt.scatter(years, totals, label="NY", c = "black", s=75)

#Set up title, axis labels, and legend, and then show:
plt.title("Lyme Disease in NY, NJ, & CT")
plt.xlabel('Years')
plt.ylabel('Occurrences')
plt.legend()
plt.show()
```

5. A popular business analytics company provides this example of A/B test:

   - A: 300 visitors, 15 conversions, and
   - B: 400 visitors, 20 conversions.

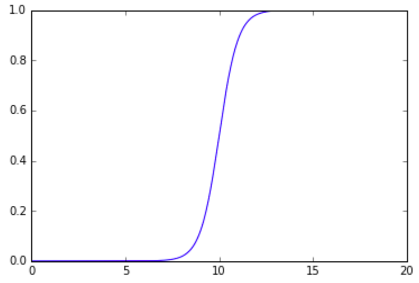
   What is the A/B test statistic for Option A and Option B (i.e. the difference of the estimated means divided by the square root of the sum of the variances)?

   Answer Key:

$$\frac{\frac{20}{400}-\frac{15}{300}}{\sqrt{\frac{15}{300}(1-\frac{15}{300})/300+\frac{20}{400}(1-\frac{20}{400})/400}} \quad = \quad \frac{\frac{5}{100}-\frac{5}{100}}{\sqrt{\frac{5}{100}(1-\frac{5}{100})/300+\frac{5}{100}(1-\frac{5}{100})/400}} = 0$$

6. You culled 10,000 311 records from the NYC Open Data and measured how many days it took to resolve the complaint lodged by the caller. Using this training data, you fit the following logistic function:

$$f(x) = \frac{1}{1+e^{20-2*x}}$$



   (a) With what probability would you expect that a call recorded 5 days ago is resolved? Explain your answer.

   (b) With what probability would you expect that a call recorded 10 days ago is resolved? Explain your answer.

(c) With what probability would you expect that a call recorded 15 days ago is resolved? Explain your answer.

(d) Say you sampled data from a single month and found that average time to resolve a complaint was 2.73 days with a standard deviation of 0.25 day. Does that fit with your prediction function above? Why or why not?

Answer Key:

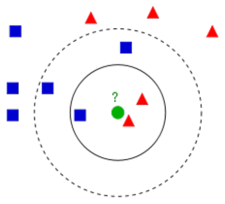
(a) `0`

(b) `50%`

(c) `100%`

(d) Our sampled data has a mean tightly clustered below 3 days, suggesting that most complaints were resolved in less than 5 days. But the logistic function predicts that complaints that occurred less than 5 days ago are not resolved. This suggests that the month sampled was an outlier or that the logistic function does not fit the data well.

7. (a) In this image from wiki (by Antti Ajanki), all points are being classified as triangles or squares. Using the k-Nearest Neighbor Algorithm for $k = 3$, what shape should be assigned to the mystery circle?



(b) Using the k-Nearest Neighbor Algorithm for $k = 5$, what shape should be assigned to the mystery circle?

(c) Write a function that takes as input:

- (x,y): coordinates of new point
- a value k, and
- a list of tuples, `(x,y,shape)`, of the location and shape of each point.

and returns the predicted shape of the new point using the k-Nearest Neighbor approach.

Answer Key: Very general approach (modified a bit) from the textbook:

```
def distance(v, w):
     """Pairwise differences"""
     return sum(abs(v_i - w_i) for v_i, w_i in zip(v,w))

def majority_vote(labels):
    """assumes that labels are ordered from nearest to farthest"""
    vote_counts = Counter(labels)
    winner, winner_count = vote_counts.most_common(1)[0]
    num_winners = len([count
                       for count in vote_counts.values()
                       if count == winner_count])
    if num_winners == 1:
        return winner                     # unique winner, so return it
    else:
        return majority_vote(labels[:-1]) # try again without the farthest

def knn_classify(k, labeled_points, new_point):
    """each labeled point should be a pair (point, label)"""
    # order the labeled points from nearest to farthest
    by_distance = sorted(labeled_points,
                         key=lambda (point, _): distance(point, new_point))
    # find the labels for the k closest
    k_nearest_labels = [label for _, label in by_distance[:k]]
    # and let them vote
    return majority_vote(k_nearest_labels)
```

8. Assume that you have that people move from the two states with the following probabilities:

75% New Yorkers stay in NY	25% of New Yorkers move to California
10% Californians move to NY	90% of CA stay in CA

which can be viewed as the Markov Chain: $\begin{pmatrix} \text{New York}_{t+1} \\ \text{California}_{t+1} \end{pmatrix} = \begin{pmatrix} 0.75 & 0.10 \\ 0.25 & 0.90 \end{pmatrix} \cdot \begin{pmatrix} \text{New York}_t \\ \text{California}_t \end{pmatrix}$

(a) Assume the initial populations are California has 40 million and New York has 20 million. What are the populations for each state:
   - in 1 year?
   - in 2 years?
   - in 3 years?

(b) Compute the eigenvalues and eigenvectors for the transition matrix:

(c) What is the steady state for the population (i.e. the same number of people move in each direction, and the populations stay the same forever)?

Hint: Think about the eigenvalues. There's a value of v_t for which $Av_t = \lambda v_t = 1v_t = v_t$.

Answer Key:

(a)
   - In 1 year, 25% of New Yorkers move to California and 10% of California residents move to New York: California: $40 + 20 \cdot \frac{1}{4} - 40 \cdot \frac{1}{10} = 40 + 5 - 4 = 41$ million.
     New York: $20 + 40 \cdot \frac{1}{10} - 20 \cdot \frac{1}{4} = 20 + 4 - 5 = 19$ million.
   - In 2 year, California: $41 + 19 \cdot \frac{1}{4} - 41 \cdot \frac{1}{10} = 41 + 4.1 - 4.1 = 41$ million.
     New York: $19 + 41 \cdot \frac{1}{10} - 19 \cdot \frac{1}{4} = 19 + 4.1 - 4.1 = 19$ million.
   - In 3 year, California: $41 + 19 \cdot \frac{1}{4} - 41 \cdot \frac{1}{10} = 41 + 4.1 - 4.1 = 41$ million.
     New York: $19 + 41 \cdot \frac{1}{10} - 19 \cdot \frac{1}{4} = 19 + 4.1 - 4.1 = 19$ million.

(b) Need to solve $A \cdot v = \lambda \cdot v$ for v and λ where A is the array. That's equivalent to $(A - \lambda) \cdot v = 0$. That is,

$$\begin{pmatrix} 0.75 - \lambda & 0.10 \\ 0.25 & 0.90 - \lambda \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 0$$

The characteristic equation is

$$\begin{aligned}
0 &= (0.75-\lambda)(0.90-\lambda)-(0.10)(0.25) \\
&= \tfrac{3}{4}\tfrac{9}{10}-\tfrac{9}{10}\lambda-\tfrac{3}{4}\lambda+\lambda^2-\tfrac{1}{10}\tfrac{1}{4} \\
&= \tfrac{27}{40}-\tfrac{36+30}{40}\lambda+\lambda^2-\tfrac{1}{40} \\
&= \lambda^2-\tfrac{66}{40}\lambda+\tfrac{26}{40} \\
&= (\lambda-1)(\lambda-\tfrac{26}{40}) \\
\lambda &= \tfrac{26}{40}, 1
\end{aligned}$$

We're interested in the largest eigenvalue, which is 1. The corresponding eigenvector is the solution to $A \cdot v = \lambda \cdot v = 1 \cdot v = v$:

$$\begin{pmatrix} 0.75 & 0.10 \\ 0.25 & 0.90 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

Or

$$\begin{pmatrix} 0.75v_1 + 0.10v_2 \\ 0.25v_1 + 0.90v_2 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

which gives $0.25v_1 = 0.10v_2$. Any choice that makes this true is the eigenvector (often choose so that the length is 1, but not necessary). So,

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2.5 \end{pmatrix}$$

(c) The steady state is 41 million and 19 million.

9. Given the points: $(1, 1)$, $(1, 3)$, and $(5, 2)$.

   (a) Compute the midpoint of the three points under the Euclidean (traditional) distance.

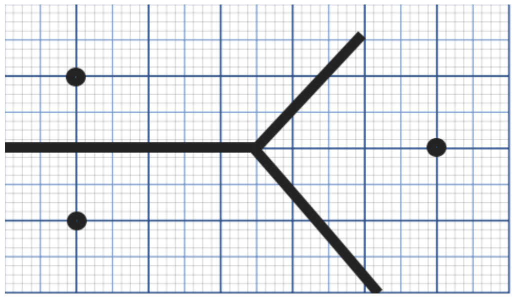
   (b) Draw the Voronoi Diagram for the Euclidean distances.

   (c) Compute the midpoint of the three points under the Manhattan distance.

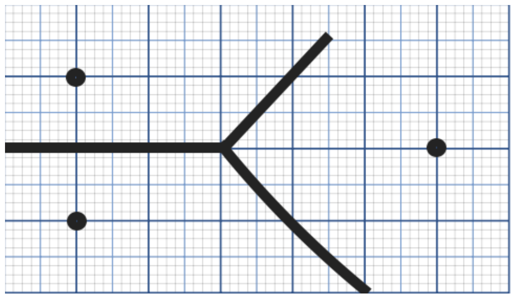
   (d) Draw the Voronoi Diagram for the Manhattan distances.

Answer Key:

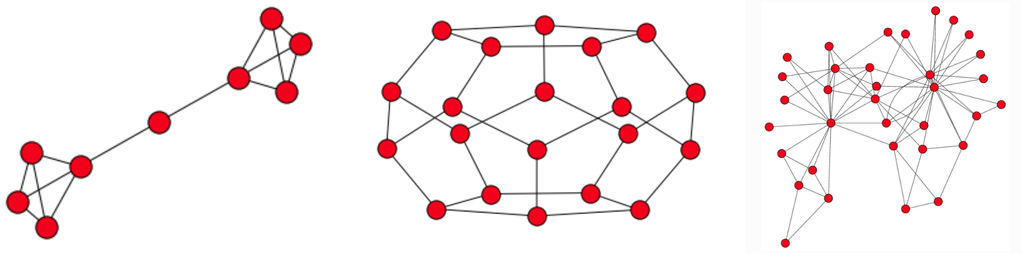
(a) Midpoint (m_1, m_2) satisfies: $(1-m_1)^2+(1-m_2)^2 = (1-m_1)^2+(3-m_2)^2 = (6-m_1)^2+(2-m_2)^2$. Using the first equations, we have $m_2 = 2$. Plugging this into the second equation, $(1-m_1)^2+1^2 = (6-m_1)^2$, $m_1 = 3.4$. The midpoint is $(3.4, 2)$.

(b) 

(c) For Manhattan distance, the distance is $|x_1 - x_2| + |y_1 - y_2|$. The distance between the first point and last is 6 (walk up 1 block and over 5 blocks). Similarly, for the second and third point. The point that's midway between all of them is $(3, 2)$ since it has Manhattan distance 3 to each of the points.

(d) 

10. In network analysis, the node most "central" to the graph plays a crucial role.



(a) There are several variations on centrality. Define what the "central" node means to you.

(b) Which node is central in each graph above? Explain your answer.

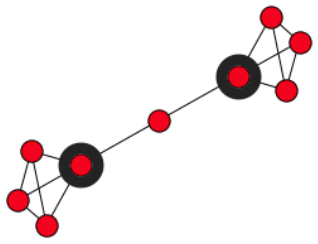
(c) Design a program that takes as input a graph, and returns the most central node:

Answer Key: *Note: there are multiple possible answers for each:*

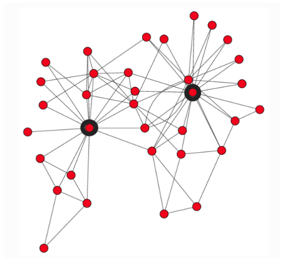
(a) There are many ways to define "central". Two common ones are:

- the node(s) with the highest degrees (that is, the most number of connections to other nodes)
- the node(s) with the most number of shortest paths that must pass through it (that is, nodes that you are most often need to visit on the way from one node to the other).

(b) Working with out first definition, nodes with highest degree are marked below:



In the middle graph, all nodes have the same degree.



(c) Pseudocode for a program that returns a list of highest degree nodes:

Inputs: Graph G

```
   i. Set Max = first node in the graph.
  ii. Set M = deg(first node in the graph)
 iii. For each node, n, in the graph G,
  iv.     If deg(n) > M:
   v.         Set Max = n.
  vi.         Set M = deg(n).
 vii.     Else If deg(n) == M:
viii.         Append n to Max.
  ix. Return Max.
```