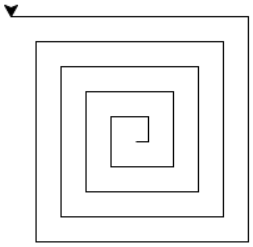
Answer Key: CMP 108/MAT 135/SOC 251: SAMPLE Final Exam, Spring 2017

1. What does the code draw:

```
import turtle
tess = turtle.Turtle()
for i in range(10,200,10):
     tess.forward(i)
     tess.left(90)
```



Answer Key:

2. What will the following Python code print:

```
b = "Apr 15, 2017"
c = b.split()
print(c)
a = ",Jan,Feb,Mar,Apr,May,Jun"
d = a.split(",")
print(d[1:4])
e = (a.find(c[0]) - 1) / 3
print(e)
f = c[1][:-1]
print(str(int(e)) + "/" + f + "/" + c[2])
```

Answer Key:

```
['Apr', '15,', '2017']
['Jan', 'Feb', 'Mar']
4.0
4/15/2017
```

3. Write a program that implements the pseudocode:

   (a) Ask the user for the number of minutes until the work day ends.

   (b) Print out the hours until the work day ends.

   (c) Print out the leftover minutes until the work day ends.

Answer Key:

```
#some comments
num = int(input('Enter number of minutes until work day ends: '))
hours = num//60
mins = num%60
print('There are', hours, 'hours')
print('and', mins, 'minutes')
```

4. (a) Write a **complete** Python program that prompts the user for a file name and prints the number of lines in the file.

Answer Key:

```
#some comments

def main():
    fileName = input('Enter file name: ')
    infile = open(fileName)
    data = infile.read()
    print("Number of lines:", data.count("\n"))

    infile.close()
```

(b) Write a **complete** Python program that prints the total 2010 population stored in a data file. Your program should open the file, `population.csv` and sum the last values in each line. The data is separated by commas (","). Your program should print the total sum that you calculated.

population.csv:

```
Borough, 2000 Population, 2010 Population
Bronx, 1332650, 1385108
Brooklyn, 2465326, 2504700
Manhattan, 1537195, 1585873
Queens, 2229379, 2230722
Staten Island, 443728, 468730
```

Answer Key: Using pandas:

```
#some comments
import pandas as pd

pop = pd.read_csv("population.csv")
sum = pop["2010 Population"].sum()

print("Total population:", sum)
```

You can also use standard file I/O:

```
#some comments

sum = 0
infile = open("population.csv")
infile.readline()   #Ignore first line, since no numbers
lines = infile.readlines()
```

```
    for l in lines:
        cells = l.split()
        sum = sum + eval(cells[2])

    print("Total population:", sum)

    infile.close()
```

5. Complete the following Python program, which sets up a graphics window and turtle, draws a hexagon (6-sided figure) to the window, and then prints a closing message and closes the graphics window when mouse is clicked. That is, write the functions `setUp()`, `drawHexagon()`, and `conclusion()`:

```
import turtle

def main():
    w,t = setUp()   #sets up a graphics window and turtle
    drawHexagon(t)  #draws a hexagon using the turtle
    conclusion(w)   #prints goodbye and closes window on click

main()
```

Answer Key:

```
def setUp():
    trey = turtle.Turtle()
    win = turtle.Screen()
    return(win,trey)

def drawHexagon(t):
    for i in range(6):
        t.forward(100)
        t.right(360/6)

def conclusion(w):
    print("Goodbye!")
    w.exitonclick()
```

6. (a) Write a function that takes number between 1 and 7 as a parameter and returns the corresponding day of the week as a string. For example, if the parameter is 1, your function should return `"Monday"`. If the parameter is 2, your function should `"Tuesday"`, etc. If the parameter is not between 1 and 7, your function should return the empty string.

Answer Key:

```
def returnDay(num):
    if num == 1:
        return "Monday"
    elif num == 2:
```

```
        return "Tuesday"
    elif num == 3:
        return "Wednesday"
    elif num == 4:
        return "Thursday"
    elif num == 5:
        return "Friday"
    elif num == 6:
        return "Saturday"
    elif num == 7:
        return "Sunday"
    else:
        return ""
```

(b) Write the Python code for the function below:

```
getInput()
   Ask user for an even number
   Until they enter an even number
      Print error message
      Ask user for an even number
   Return the even number entered
```

Answer Key:

```
def getInput()
    x = int(eval('Enter an even number: '))
    while x % 2 != 0:
        print('Not an even number!')
        x = int(eval('Enter an even number: '))
    return(x)
```

7. The file `nycHistPop.csv` contains historical population data for the boroughs of New York City. The first couple of lines of the file are:

```
Year,Manhattan,Brooklyn,Queens,Bronx,Staten Island,Total
1698,4937,2017,,,727,7681
1771,21863,3623,,,2847,28423
1790,33131,4549,6159,1781,3827,49447
```

(a) Modify the following program to plot the percentage of New Yorkers that live in the Bronx:

```
import matplotlib.pyplot as plt

import pandas as pd
pop = pd.read_csv('nycHistPop.csv')
pop.plot(x="Year")

plt.show()
```

Answer Key:

```
import matplotlib.pyplot as plt

import pandas as pd
pop = pd.read_csv('nycHistPop.csv')

pop['FractionBronx'] = pop['Bronx']/pop['Total']
and then can use it to create a new graph:
pop.plot(x = 'Years', y = 'FractionBronx')

plt.show()
```

(b) Given the program above, fill in the code that will:

i. Print out the maximum number of people living in the Bronx:

Answer Key:

```
print("The largest number living in the Bronx is", pop["Bronx"].max())
```

ii. Print out the number of years of data in the file:
(Hint: Each year is stored in a separate row)

Answer Key:

```
print("The number of years of data is ", pop["Bronx"].count())
```

iii. Make a bar plot instead of a line graph:

Answer Key:

```
pop.plot.bar(x="Year")
```

8. What will the following R code print:

```
> poker_vector <- c(140, -50, 20, -120, 240)
> poker_vector[1]
> days_vector <- c("Monday", "Tuesday",
   "Wednesday", "Thursday", "Friday")
> names(poker_vector) <- days_vector
> poker_vector("Tuesday")
> min(poker_vector)
# Which days did you make money on poker?
> selection_vector <- poker_vector > 0
> poker_vector[selection_vector]
```

Answer Key:

```
140
-50
```

```
-120
Monday Wednesday    Friday
      140        20       240
```

9. Fill the R code that will do the following:

   (a) Create a vector, `temps`, of the high temperatures in New York City in March 2017:

   ```
   >                        (66,63,38,29,35,44,50,59,60,47,28,30,35,32,27,39,47,38,47,
                             51,59,49,43,55,56,42,50,46,58,51,43)
   ```

   Answer Key:

   ```
   >   temps <-  c(66,63,38,29,35,44,50,59,60,47,28,30,35,32,27,39,47,38,47,
                             51,59,49,43,55,56,42,50,46,58,51,43)
   ```

   (b) Print the average high temperature recorded over the month:

   ```
   >
   ```

   Answer Key:

   ```
   >   mean(temps)
   ```

   (c) Create a new vector, `runningMax`, with the running maximum over the month:

   ```
   >   runningMax <-
   ```

   Answer Key:

   ```
   >   runningMax <-  cumax(temps)
   ```

   (d) Make a plot of the data stored in `runningMax`

   ```
   >
   ```

Answer Key:

```
> plot(runningMax)
```

10. Write a program that prints out the correlation table and computes the chi-squared test on the hypotheses that fertility is independent of the other variables in the built-in data set, `swiss`. The data set `swiss` contains standardized fertility measure and socio-economic indicators for each of 47 French-speaking provinces of Switzerland at about 1888. The structure of the data set is:

```
> str(swiss)
'data.frame': 47 obs. of  6 variables:
 $ Fertility       : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4 82.9 ...
 $ Agriculture     : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3 45.2 ...
 $ Examination     : int  15 6 5 12 17 9 16 14 12 16 ...
 $ Education       : int  12 9 5 7 15 7 7 8 7 13 ...
 $ Catholic        : num  9.96 84.84 93.4 33.77 5.16 ...
 $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21 24.4 ...
```

Answer Key:

```
cor(swiss)
chisq.test(swiss$Fertility, swiss$Agriculture)
chisq.test(swiss$Fertility, swiss$Examination)
chisq.test(swiss$Fertility, swiss$Education)
chisq.test(swiss$Fertility, swiss$Catholic)
chisq.test(swiss$Fertility, swiss$Infant.Mortality)
```

Another approach:

```
cor(swiss)
for (n in names(swiss) {
  y <- swiss[n]
  print(chisq.test(swiss$Fertility, y))
}
```