1. Write a code segment that creates a window that is 300 pixels wide and 400 pixels tall and then draws a rectangle with one corner at location (50, 30) and its diagonally opposite at location (150, 200). The user should then be prompted (from the Python shell) to strike any key to close the window. Assume the graphics package has been imported with the directive from graphics import *

2. Append to the following segment of code additional lines that:
   - Will prompt the user (in the shell) for the number of mouse clicks to expect from the user. Assume it will be an integer greater than 1.
   - Upon receiving the first click from the user the program will draw a point at the location where the user clicked.
   - Upon receiving additional clicks from the user the program will draw a line from the location where the user clicked previously clicked to the latest location where the user clicked. This continues until the number of mouse clicks specified in the user input have been received. When the drawing is done the user should see a collection of connected line segments in the window.
   - Upon receiving one additional mouse click the window will close.

```python
from graphics import *
from math import *

def main():
    win = GraphWin("CSCI220",400,200)
```
3. Write a segment of code that opens the file Test2-3.in and determines the average number of words per line in the text.

4. Write a segment of code that opens the file Test2-4.in and copies every odd numbered line to Test2-4.out.

5. Write a conditional statement that prints the appropriate message based on user input. Assume the user input is stored in a variable num and is an integer.

<table>
<thead>
<tr>
<th>User Input</th>
<th>Appropriate Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or less</td>
<td>“Error”</td>
</tr>
<tr>
<td>1..10</td>
<td>“Good”</td>
</tr>
<tr>
<td>11..15</td>
<td>“OK”</td>
</tr>
<tr>
<td>16 or more</td>
<td>“Awful”</td>
</tr>
</tbody>
</table>
6. Write a function `changeName(name)` that takes a string value `name` as an argument, where name is in the form

   “Last, First Middle”

for example “Anderson, Paul Anderson”. Note the use of the comma. The function should return a string that has the name in the form.

   “First Middle Last”

for example “Paul Edward Anderson.”

7. Write a function `changeName2(nameList)` that takes a list `nameList` as an argument that contains a person’s name, where name is in the form

   `[Last, First, Middle]`

for example `[‘Anderson’, ‘Paul’, ‘Edward’]`. The function should not return a value, but instead should modify the list so the name is in the form

   `[First, Middle, Last]`

for example `[‘Anderson’, ‘Paul’, ‘Edward’].`
8. Write a function `findValidValue()` → value that returns a valid value entered by the user. The function should ask the user to enter a number until the user enters a value that is any of the following.
   a. Inclusively within the range of 5 and 17.
   b. Exactly equivalent to 100.
   c. Exclusively within the range of 20 and 25.

8. Evaluate the following Boolean expressions:

   -3 < 7

   3 > 20 and 30 >= 20

   20 == 20

   7 == 8 and not(20 < 18)

   3 > 4 or (5 <= -3 and 8 == 7)

   (10 < 5 and 40 > 20) or 18 >= 10

9. Write a truth table for the following expressions:

   a. A or not(B)

   b. A and (B and C)
10. Consider the following code segment.

```python
a, b, c = eval(input("Enter three numbers: "))
if a > b:
    if b > c:
        print("Spam Please!")
    else:
        print("It's a late parrot!")
elif b > c:
    print("Cheese Shoppe")
    if a >= c:
        print("Cheddar")
elif a < b:
    print("Gouda")
elif c == b:
    print("Swiss")
else:
    print("Trees")
    if a == b:
        print("Chestnut")
    else:
        print("Larch")

print("Done")
```

Show the output that would result from each of the following possible inputs:
(a) 3,4,5

(b) 3,3,3

(c) 5,4,3

(d) 3,5,2
(e) 5,4,7

(f) 3,3,2

(g) Rewrite the code so that it does NOT use nested if-else statements (switch to elif clause).
11. Write a program that computes the fuel efficiency of a multi-leg journey. The program will first prompt for the starting odometer reading and then get information about a series of legs. For each leg, the user enters the current odometer reading and the amount of gas used (separated by a space). The user signals the end of the trip with a blank line. The program should print out the miles per gallon achieved on each leg and the total MPG for the trip.