1. Fix this code so it runs correctly.

```
def divider x, y :
     value == 23
     if y != 0:
          print("error: divide by zero)
     else
          x/y
     return value
```

2. Do a substitution trace for the following function.

```
def hanoi(n):
     if n == 1:
          return 1
     else:
          return 2 * hanoi(n-1) + 1
```

hanoi(4) =

3. What will be the value of the following statements?

```
luck = "Good Luck!"
walk = "Walk in the Park"
x=4
y=9
```

a. `luck[8]`

b. `luck[x + 6]`

c. `len(luck)`

d. `luck[y]`

e. `luck[-1]`

f. `len(walk)`

g. `walk[-len(walk)]`

h. `luck[:3]`

i. `luck[5:]`

j. `luck[6:y]`

k. `luck[3:3]`

l. `walk[::2]`

m. `luck[::-1]`

n. `luck[-2:2]`

o. `luck[-5:-2]`

4. Using the following imports, show how you would make the turtle move forward the square root of 10.

```
import math
from turtle import *
```

5. Show an illustration of what the following program draws.

```
def mystery(x):
     if x > 0:
          if x % 2 == 0:
               forward(20)
               right(90)
          else:
               back(40)
               left(90)
          mystery(x-1)
mystery(4)
```

6. Show the output of the following code snippets.

a.)

```
sum = 1
i = 0
while i < 10:
     sum = sum + i
     i = i + 1
print("sum=", sum, "i=", i)
```

b.)

```
i = 10
while i < 10:
     i = i - 1
print(i)
```

c.)

```
i = 0
s = "hello"
for ch in s:
     print("i=", i, ",ch=", ch)
     i = i + 1
```

7. Write a program that reads a file, `words.txt`, that has one word per line and prints out the total number of characters and the total number of lines in the file.

8. A palindrome is a word that is spelled the same backward and forward. For example, ‘`noon`’ and ‘`redivider`’. Recursively, a word is a palindrome if the first and last letters are the same and the middle is a palindrome.

a.) Write a boolean function, `isPalindrome`, which takes a string and returns a boolean indicating whether the string is a palindrome or not. It must use recursion.

b.) Do a complete substitution trace for: `isPalindrome('racecar')`

c.) Write an iterative version, `isPalindromeIter`, which performs the same task, but does not use recursion.

9. Sorting:
   (a) Run insertion-sort on this list, [3, 2, 4, 5, 1]

10. Tuples:
   (a) What is wrong with the following?

```
data = [1,2,3]
data[0] = 3
data[-1] = 1
print(data)
data = (1,2,3)
data[0] = 3
data[-1] = 1
print( data )
```

   (b) What is the type and value of myVar in the following?

```
def myFunc(x):
    y = x + x
    z = x * x
    return y, z

myVar = myFunc( 5 )
```

   (c) What are the types and values of foo and bar?

```
foo, bar = myFunc( 5 )
```

11. Binary Search

Tell which values would be checked if the following binary searches were done on the following list:

lst = [0, 1, 4, 6, 7, 18, 20, 34, 39, 42, 50, 72, 77, 80, 99]

(a) binarySearch(lst, 18)

(b) binarySearch(lst, 34)

(c) binarySearch(lst, 99)

(d) binarySearch(lst, 77)

(e) binarySearch(lst, 999)

12. Lists: Give the output of the following statements

(a)
```
names = [“Amir’, ‘Barry’, ‘Charles’, Dao’]
print names[-1][-1]
```

(b)
```
names1 = ['Amir', 'Barry', 'Chales', 'Dao']
names2 = names1
names3 = names1[:]

names2[0] = 'Alice'
names3[1] = 'Bob'

sum = 0
for ls in (names1, names2, names3):
   if ls[0] == 'Alice':
      sum += 1
   if ls[1] == 'Bob':
      sum += 10

print sum
```

(c)
```
names1 = ['Amir', 'Barry', 'Chales', 'Dao']
names2 = [name.lower() for name in names1]

print names2[2][0]
```

(d)
```
numbers = [1, 2, 3, 4]
numbers.append([5,6,7,8])

print len(numbers)
```

(e)
```
list1 = [1, 2, 3, 4]
list2 = [5, 6, 7, 8]

print len(list1 + list2)
```

(f)
```
myList = [ [ [0, 1, 2], [3, 4, 5] ], [6, [7, 8], 9], 10]
print(myList[0][1][2])

print(myList[1][1][1])

print(myList[2][2])
```

13. Classes and Maker Functions
(a) Design a class called 'Pet' which can store the name, age, and species of the pet

(b) Give the maker function for this class

13. Sample Practical Exam question.

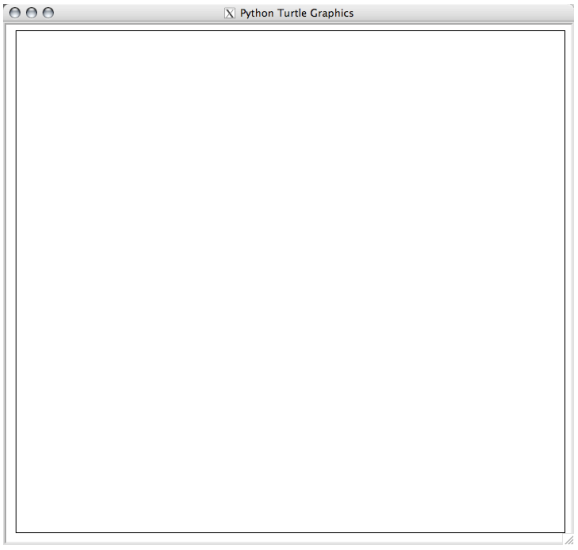
Given the following Turtle pseudocode:

```
function drawSquares(depth, length):
    if depth <= 0:
        do nothing
    else:
        move forward length
        turn left 90°

        move forward length S
        turn left 90°
        drawSquares(depth-1, length/3)

        move forward length S
        turn left 90°

        move forward length S
        turn left 90°

maxsize = 1 00
read the depth from input
call drawSquares(depth,  maxsize)
wait for input before exiting
```

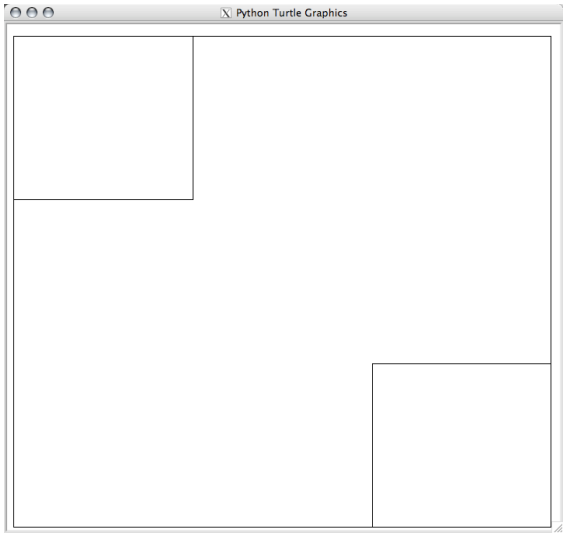
a) Implement the pseudocode in Python in the file `unosquares.py`. Your programs do not have to match the exact size and starting position on the canvas

b) Make a copy of your program and call it `duosquares.py`. Modify the implementation to draw the image on the following page.

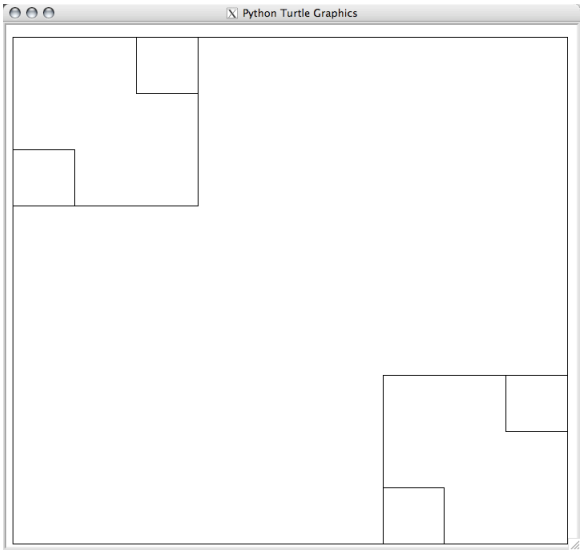
Depth = 0:



Depth = 1:



Depth = 2:



Depth = 3: