1. Write a function which reverses a string (e.g. “Don’t get sick” becomes “kcis teg t’noD”).

```
1 def reverse( string ):
2     if string == '' :
3         return string
4     else:
5         return reverse( string[1:] ) + string[0]
```

2. Define a function that takes an input string and rotates the sequence of letters in each word by n. For example: shift_left(“DEADBEEF”, 3) will produce the output string “DBEEFDEA”. You should be able to shift a string by a value greater than the length of the string[1].
*Assume a function len( str ) which returns the length of a string is provided*

   (a) Design: Give brief description on how your function should accomplish this.
   Our implementation will grab the first part of the new string by slicing all characters at an index after the offset. The second part of the string will be all of the characters up to the offset point. We will then concatenate the two strings.
   Making it possible for the offset to be greater than the length of the string can be accomplished by making `offset = offset mod len(string)`

   (b) Testing: Provide 3 test cases, using specific values for the input string and amount of shifting and what the expected output should be for each.
   shift_left(“”, 5) should return ””
   shift_left(“A”,5) should return ”A”
   shift_left(“ABADCAFE”,300) == shift_left(”ABADCAFE”,4)
   shift_left(“FOO”, 1) should return ”OOF”

   (c) Implement the function in Python.

```
1 def shift_left(string, offset):
2     if len(string) == 0:
3         return string
4     else:
5         offset = offset % len(string)
6         first = string[offset:]
7         last = string[:offset]
8         return first + last
```

   (d) Implement the function `shift_right()`, which rotates letters in the opposite direction.

```
1 def shift_right(string, offset):
2     return shift_left(string, -offset)
```

[1]The % operator, which finds the remainder of a division operation will be useful here.

3. Write a function that takes in a file name, and returns the average size of a word in that file. The files will only have 1 word per line, for example:

No
soup
for
you!

which has an average length of: 3.25
*Assume a function len( str ) which returns the length of a string is provided*

```
def average_wordlength(filename):
    characters = 0
    words = 0
    for line in open(filename):
        words += 1
        characters += len(line)
    return characters/words
```

4. Perform a substitution trace on

```
reverse('Cinco-fone')
```

```
reverse('Cinco-fone')
reverse('inco-fone') + 'C'
reverse('nco-fone') + 'i' + 'C'
reverse('co-fone') + 'n' + 'i' + 'C'
reverse('o-fone') + 'c' + 'n' + 'i' + 'C'
reverse('-fone') + 'o' + 'c' + 'n' + 'i' + 'C'
reverse('fone') + '-' + 'o' + 'c' + 'n' + 'i' + 'C'
reverse('one') + 'f' + '-' + 'o' + 'c' + 'n' + 'i' + 'C'
reverse('ne') + 'o' + 'f' + '-' + 'o' + 'c' + 'n' + 'i' + 'C'
reverse('e') + 'n' + 'o' + 'f' + '-' + 'o' + 'c' + 'n' + 'i' + 'C'
'e' + 'n' + 'o' + 'f' + '-' + 'o' + 'c' + 'n' + 'i' + 'C'
'enof-ocniC'
```

5. Write a function that takes in a string representation of a number and returns the sum of all of digits in the string.
   For example sum( '11111' ) returns 5.
   *Assume a function len( str ) which returns the length of a string is provided.*
   *assume a function int( str ) which, given a string representation of an integer, returns its integer value*

   (a) Recursively.

```
1 def sumRec( numbers ):
2     if len( numbers ) == 0:
3         return 0
4     else:
5         return int(numbers[0]) + sumRec( numbers[1:] )
```

   (b) Iteratively

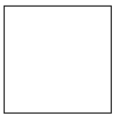
```
1 def sumIt( numbers ):
2     total = 0
3     while numbers != '':
4         total += int( numbers[0] )
5         numbers = numbers[1:]
6     return total
```

   (c) How would you test this function?

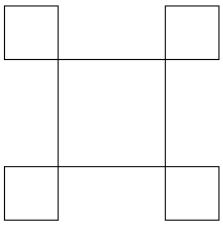
   - Empty string
   - String length = 1
   - string length > 1

6. Assuming the turtle is facing east, write the python code to draw the following picture given the proper depth as input:

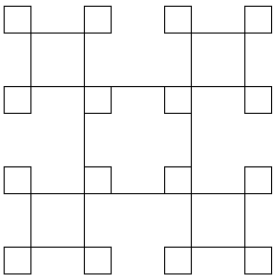
   - depth = 0
     No output
   - depth = 1

   - depth = 2

   - depth = 3

```
1         def drawSqaures( length, depth ):
2             if depth <= 0:
3                 return
4             count = 4
5             while count > 0:
6                 turtle.forward( length )
7                 turtle.left( 90 )
8                 drawSqaures( length/2, depth-1 )
9                 turtle.right( 180 )
10                count -= 1
```

7. What does the following evaluate to?

```
1 def writeThatDown( n ):
2     if n < 5:
3         return n
4     return (2 * n)
5
6 def he( n ):
7     temp = n + 180
8     if temp > 185:
9         return temp
10    return n
11
12 def putstheFernback( n ):
13    return -n
14
15 n = 20
16 n = he(putstheFernback(writeThatDown( n ) ) )
17 print( n )
```

-40

8. Write a function which performs a basic string compression, which uses counts of repeated characters. For example the string *abbbccccaaa* would be compressed to: *a*1*b*3*c*4*a*3. *Assume a function len( str ) which returns the length of a string is provided.*
*assume a function int( str ) which, given a string representation of an integer, returns its integer value.*
*Assume a function str( int ) which, given an integer, returns the string representation of this integer, is provided.*

```
1 def compress( string ):
2     new = ''
3     curChar = string[0]
4     count = 0
5     for char in string:
6         if char == curChar:
7             count += 1
8         else:
9             new = new + curChar + str(count)
10            curChar = char
11            count = 1
12
13    new = new + curChar + str(count)
14    return new
```