CSCI-141 Exam 1 Review

September 21, 2017

Presented by the RIT Computer Science Community

http://csc.cs.rit.edu

1. Python basics

   (a) Write an iterative function to add up all numbers from 1 to n (inclusive) and return the sum.

```
def foo_iter(n):
    total = 0
    for num in range(1,n+1):
        total += num
    return total
```

   (b) Re-write the function above using recursion. Is your answer tail recursive? Why?

   The answer described below *is* tail recursive because no additional information must be stored on the stack for each recursive call.

```
def foo_rec(n, count=1, total=0):
    if count > n:
        return total
    total += count
    count += 1
    return foo_rec(n, count, total)
```

   (c) How would you test the two functions you've written above? Explain test cases you would want to analyze, and write a function to run the tests.

   - Extraneous cases: n=None, n is not an integer
   - Base cases: $n = 0$, $n = 1$
   - General case: $n > 1$

   The above functions are fruitful (have return values), so we can write test cases using the expected outputs. This can be done with hard-coded values or using a separate algorithm to assert the validity of our function's return values.

```
def test_foo_functions(n):
    # below suggests the formula for the sum 1+2+...+(n-1)+n
    expected = ( n * ( n + 1 ) ) // 2
    if foo_iter(n) != expected:
        print("iterative foo function produced an incorrect result!")
    if foo_rec(n) != expected:
        print("recursive foo function produced an incorrect result!")
```

2. Write a **recursive function** that reverses a string (e.g. "Racecar" yields "racecaR").

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

Other solutions are possible.

3. Perform a substitution trace on `reverse('Doge')`.

```
reverse('Doge') =   reverse('oge') + 'D'
                =   (reverse('ge') + 'o') + 'D'
                =   ((reverse('e') + 'g') + 'o') + 'D'
                =   (((reverse('') + 'e') + 'g') + 'o') + 'D'
                =   (((''  + 'e') + 'g') + 'o') + 'D'
                =   'egoD'
```

4. 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

5. Define a function that takes an input string and rotates the sequence of letters in the string by n. For example: shift_left("DEADBEEF", 3) will produce the output string "DBEEFDEA".
shift_left("Giant Robot", 4) will produce "t RobotGian"
shift_left("X", 5) will produce "X"
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 (%) len(string)`.

   (b) Testing: Provide at least 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. Use `shift_left()` in your implementation [2].

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

[1] The modulus (%) operator, which finds the remainder of a division operation, will be useful here.
[2] The behavior of % with negative numbers varies by language. In Python, it works like this: 7%2=1 and -7%2=1.

6. Write a function that takes in a file name, and returns the average size of a word in that file. Assume 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.*

```
1 def average_wordlength(filename):
2     characters = 0
3     words = 0
4     for line in open(filename):
5         words += 1
6         characters += len(line)
7     return characters/words
```

7. 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( '11235' )` returns 12.
Assume a function `len( str )` *that returns the length of a string is provided.*
Further, assume there is a function `int( str )` *which, given a string representation of an integer, returns its integer value.*

   (a) Recursively.

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

   (b) Iteratively

```
1 def sumIt( numberStr ):
2     total = 0
3     for c in numberStr:
4         total += int( c )
5     return total
```

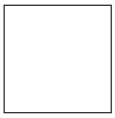
   Other solutions are possible.

   (c) How would you test this function?

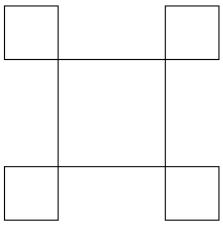
   - Empty string
   - String length = 1
   - String length > 1

8. Assuming the turtle is facing East, write the Python code to draw the following picture given the proper depth as input:

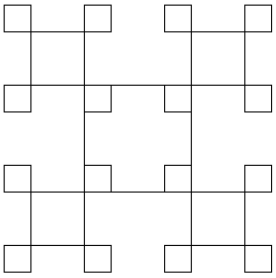
   - depth = 0
     No output
   - depth = 1



   - 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
```

NOTE: Solution starts from top-left corner of the main box.

9. Connor is a big dweeb and loves keeping a count of things. You will be writing a tail-recursive function to satisfy his desires.
Assume you are given the functions strHead() and strTail(), which run in constant time. strHead() returns the first character in the string and strTail() returns the rest of the string.

   (a) Write a tail recursive function `coRec`, which takes a string and a character and returns the number of times the character appears in the string.
   For example, `coRec("Eric is enjoying the weather.", "i")` should return `3`.
   Do not use the `len()` function.

```
def coRec(string,char,counter=0):
    if(string==""):
        return counter
    if(strHead(string)==char):
        return coRec(strTail(string),char,counter+1)
    else:
        return coRec(strTail(string),char,counter)
```

   (b) What is the complexity of the functions you wrote for 9a?
   It runs in O(N) time.

10. Write a function which performs basic string compression, using the counts of repeated characters. For example, the string *abbbccccaaa* would be compressed to: *a*1*b*3*c*4*a*3.
Assume the function `len( str )`, which returns the length of a string, is provided. Further, assume you may use `int( str )` and `str( int )`, which converts a string to its integer representation, and converts an integer to its string representation, respectively.

```
def compress( string ):
    new = ''
    curChar = string[0]
    count = 0
    for char in string:
        if char == curChar:
            count += 1
        else:
            new = new + curChar + str(count)
            curChar = char
            count = 1

    new = new + curChar + str(count)
    return new
```

11. Although most in the industry will be reasonably forgiving when it comes to coding by hand, you should still be comfortable with the syntax of your language of choice.

    (a) Identify and fix the line(s) that have invalid syntax in the following code:
    If you are unsure about a specific line, just write a brief explanation of the error. Assume that incorrect lines do not affect the validity of other lines which may depend on them.

```
1  include math
2
3  int a = 0
4  b = 0
5
6  def function(arg):
7      return (arg + 2)
8
9  c = "hello"
10 sum = b + math.sqrt(c)
11 total = a + (b ** 0.5)
```

    1 This line should read "`import math`", or, alternatively, "`from math import *`".

    3 Python is a dynamically-typed language, so you cannot specify a variable's type in this manner.

    10 This line will cause a runtime error because `sqrt` tries to perform mathematical operations on a string, which is invalid.

    (b) Which of the following *are* keywords or standard functions in Python?

```
import     in     from     class     define     int     str     String
True     false     where     while     or     not     isinstance     new
print     range     float     char     bool     xor     sum
```

    All of these are valid, except `define, String, false, where, new, xor, char`.

12. Write a function that takes a filename as its parameter and prints all the last names from the given file. Each line of the file follows the format of a first name, a space, a last name, a space, and miscellaneous extra data.

```
1 def printLastNames(fileName):
2     with open(fileName) as file:
3         for line in file:
4             line = line.strip()
5             fields = line.split()
6             print(fields[1])
```