CSCI 101 Final Exam Review

The final exam will be cumulative but emphasize material from the second half of the course. All course topics are fair game; expect in-depth questions on topics since the midterm – number representation (hexadecimal and two's-complement), circuits, architecture, assembly language, lists, loops, dictionaries, objects, complexity and algorithms – that build on topics from the first half of the course.

1. Review all posted notes, slides, examples, and sample solutions on the course web site. Review your own homework assignments and midterm.

2. Lists
   a) Write a Python function named indexMin that takes a list of integers as a parameter and returns the index of the smallest integer in the list. (If there are duplicate list entries your function should return the index of the first occurrence of the smallest value.) You may assume that there is at least one integer in the list. Example: indexMin([34, 25, 67, 12, 90, 12]) should return 3.

   b) What output is produced by the following code?

   ```python
   def mystery(a):
       print(a)
       for i in range(1, len(a)):
           a[i] += a[i-1]
       print(a)
   mystery([8, 5, 0, -7, 4])
   ```

3. Loops and nested loops
   a) The "Hailstone" sequence starts with any positive integer n, and defines the next value in the sequence as follows: if n is even, the next value is n/2; if n is odd, the next value is 3n+1. (The Collatz conjecture states that for any starting value of n, the sequence will always converge to 1. Although no counterexample is known, the conjecture has not been proven.) For example, for a starting value of n=10, the sequence is 10, 5, 16, 8, 4, 2, 1. Write a non-recursive function hailstone(n) that implements the Hailstone function and prints the sequence of values for an input n until the sequence reaches 1. For example, hailstone(10) should produce 10 5 16 8 4 2 1.

   b) Rewrite the following function to use a while loop instead of a for loop.

   ```python
   def sumOfList(t):
       """ Returns the sum of all values in list t."""
       sum = 0
       for v in t:
           sum += v
       return sum
   ```

c) Consider the following function:
Note: recall that end='' in the print statement means that it won’t automatically go to the next line after printing the symbol. Also, print() with no arguments just prints a newline character so that the next thing to be printed will be on the next line.

def printSquare(count, symbol):
    for i in range(count):
        for j in range(count):
            print(symbol, end=' ')
        print()

1. What would be printed out by printSquare(5,'#')?

2. Write a new version of the function called printTriangle(count, symbol), that prints out a right triangle of the same dimensions as the square. E.g., a sample run would be:

   >>> printTriangle(5,'#')
   #
   ##
   ###
   ####
   #####

4. Dictionaries
What output is produced by the following code?

d = dict()
d['summer'] = 'Sommer'
d['fall'] = 'Herbst'
for x in ['spring', 'summer', 'fall', 'winter']:
    if x in d:
        print(d[x])
    else:
        print(x)

5. Objects
   a) When writing object-oriented Python code, what does self refer to?
   b) What is the difference between objects and classes? What is the relationship between them?

6. Algorithms and Complexity
   a) Give a brief description of the insertion sort algorithm. Express in Big-O notation the worst-case running time of insertion sort on n integers. Which sorting algorithm has a significantly better run-time?
   b) Suppose you have a sorted list of 2000 items. How many comparisons would you need to make (in the worst case) in order to find out whether a particular given value appears in the list or not?
   c) Suppose that an algorithm that is O(n) takes 15 seconds on your current computer to solve a problem in which n is 1,000. About how long will it take your computer to solve the problem when n is 3,000? Explain briefly.
   d) What is the worst case running time (expressed using Big-O notation) to sort a list of n integers using merge sort?

7. (a) Add the two 8-bit two's-complement binary numbers 11100111 and 00011111 together in binary, showing all your work. (b) Convert D7 from two’s complement hexadecimal to decimal.
8. Draw a circuit diagram that implements the XOR function (for two inputs) using only AND, OR, and NOT gates. [Recall that XOR is true when exactly one of its inputs are true, but not both.]

9. Write an HMMM assembly function PrintN2. The function should read in one input, a number n. Your function should square n, and then count down (output) all the numbers from n^2 to 1. Example:

Enter number: 3
9
8
7
6
5
4
3
2
1

10. Write a recursive function countEven(t) in Python that takes list t as an input parameter and returns the number of even numbers in t. Example: countEvens([2,5,7,6,4]) should return 3.

11. The explode() function takes in a string s as a parameter and returns a list of characters in s. For example, explode('fantastic') would return ['f','a','n','t','a','s','t','i','c']. A recursive version is below. Rewrite this using a for loop instead.

    def explode(s):
        """Returns list of characters in string s."""
        if len(s) == 0:
            return []
        else:
            return explode(s[:-1]) + [s[-1]]

12. Without looking at the posted code, write the function Koch(len, levels) to draw the recursive Koch curve as shown on right below. If levels is 0, the turtle should simply go forward by len; in all other cases, the turtle should also end up at the same position.

13. Consider the following mystery function.

    def mystery(t, x):
        for v in t:
            if v == x:
                return True
        return False

    a) What values would be returned from the calls
       i. mystery(['a','b','c','d'])?
       ii. mystery(['a','b','c','d'])?

    b) Describe in English what the mystery function does.