CSCI 101 Midterm Sample Questions

Note: you may bring one 8.5"x11" double-sided sheet of notes for your use during the exam (hand-written or typed). Otherwise, no notes, computers, calculators, phones or other aids are allowed.

1. *Short answer*

(a) The following function is supposed to test whether a given word contains a given letter. It uses a recursive strategy. Unfortunately, there are three different problems with this function. Identify these problems and rewrite the corrected function.

```
def contains(word, letter):
    if word[0] == letter:
        return True
    elif len(word) = 0:
        return False
    else:
        contains(word[1:], letter)
```

```
def contains(word, letter):
  if len(word) == 0:
    return False
  elif word[0] == letter:
    return True
  else:
    return contains(word[1:], letter)
```

error 1: order of base cases needs to be switched

error 2: if len(word) = 0 needs == instead of =

error 3: last line is missing the word 'return'

(b) What output do the following lines of Python produce?

```
x = 10
y = x * 3 + 5
x = 2 * y – 5
print("x=", x)
print("y=", y)
```

x = 65
y = 35

(c) Describe what makes a problem suitable for a recursive solution. What are the necessary components or characteristics of a recursive solution?

A problem might be suitable for a recursive solution if the solution to the entire problem can be expressed in terms of the solution to a smaller problem. For example, N! = N * (N-1)! The two necessary components of a recursive solution are (1) a base case that solves a trivial version of the problem, and (2) a recursive case that solves a smaller version of the problem while making progress towards a base case.

2. *True or False*

For each of the statements below state whether they are true or false. No justification required.

False If `s='eat more kale'` then `s[1] == s[-1]` would give `True`.

True Given an even-length string s, `s[(len(s)//2):]` would give the last half of the string.

True `7/2 + 7//2 + 7%2` evaluates to `7.5`

False Given the function below, if we were to type `print(identity(3))` and hit enter in the Python console, we would only see `3` displayed.

```
def identity(num):
    print(num)
    return num
```

False The function `rundown(n)` produces the even numbers from n down to 2.

```
def rundown(n):
    if n>1:
        print(n)
        rundown(n//2)
```

3. *Understanding code*

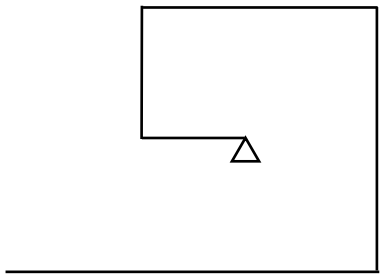
(a) Consider the following Python program. What output will this code produce when run?

```
def mystery1(a, b):
    return a+b

def mystery2(a, b):
    if a > b:
        return a
    else:
        return b

def mystery3(a, b):
    if a < b:
        print(a)
        a = a + 2
        mystery3(a, b)

y = mystery1(26, 8)
print(y)

y = mystery2(26, 8)
print(y)

mystery3(16, 22)
```

34
26
16
18
20

(b) Consider the following function definition.

```
def mystery(side):
    if side > 0:
        turtle.forward(side)
        turtle.left(90)
        mystery(side-50)
```



Draw what the function above would draw on the screen if run with command `mystery(250)`. Assume the turtle initially is at the center of the window facing right. You can assume the width and height of the window are about 600 pixels. Indicate the final position and orientation of the turtle with a small triangle.

4. Writing functions

(a) Complete the Python program **`midterm1.py`**. The program asks the user for input, then calls functions **`test1`** and **`test2`** and outputs their results. You should define the (non-recursive) functions **`test1`** and **`test2`** so that the provided main program (at the bottom of the box below) will work without any modification. Do not use any pre-defined mathematical functions, rather simply use multiplication, addition, conditionals, etc, to complete the code. Here is an example of how your program should work (the user's input is the -7 in bold). Note: The function calls and input/output are done for you; you do not need to rewrite the main program, just write the function definitions for **`test1()`** and **`test2()`**.

```
>>> runfile('midterm1.py')
Enter a number: -7
square of -7 is 49
absolute value of -7 is 7
```

```
# midterm1.py

# define functions test1 and test2 here:

  def test1(x):
    return x * x

  def test2(x):
    if x < 0:
      return -x
    else:
      return x


# main program that calls test1 and test2:

x = int(input("Enter a number: "))
print("square of", x, "is", test1(x))
print("absolute value of", x, "is", test2(x))
```

(b) Write a recursive function called `every_other_letter` that takes a string parameter as input and returns a string that contains every other letter in the string starting with the first letter. For example:

```
>>> every_other_letter('banana')
'bnn'
>>> every_other_letter('computer')
'cmue'
>>> every_other_letter('kale')
'kl'
>>> every_other_letter('a')
'a'
```

```
def every_other_letter(s):
  if len(s) < 2:
    return s
  else:
    return s[0] + every_other_letter(s[2:])
```

[Note: a concise correct answer can be given in just 5 lines of code. Just give the function definition, not the main program, and do not use **`print()`** or **`input()`**.]

5. *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.

```
def indexMin(t):
  minIndex = 0
  for i in range(1,len(t)):
    if t[i] < t[minIndex]:
      minIndex = i
  return minIndex
```

b) What output is produced by the following code?

```
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])
```

```
[8, 5, 0, -7, 4]
[8, 13, 0, -7, 4]
[8, 13, 13, -7, 4]
[8, 13, 13, 6, 4]
[8, 13, 13, 6, 10]
```

6. *Loops*

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

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

```
def sumOfList(t):
  sum = 0
  i = 0
  while i < len(t):
    sum += t[i]
    i += 1
  return sum
```

7. 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.

```
def countEven(t):
  if len(t) == 0:
    return 0
  else:
    return (((t[0] % 2) + 1) % 2) + countEven(t[1:])
```

8. 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]]
```

```
def explode(s):
    t = []
    for c in s:
        t += [c]
    return t
```

9. Consider the following mystery function.

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

mystery(['a','b','c'],'d'): False
(because 'd' is not in the input list)

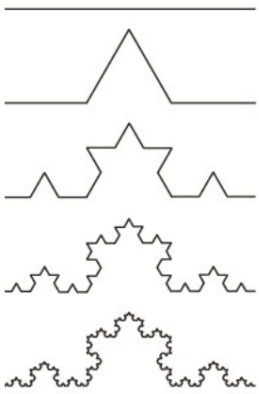
mystery(['a','b','c'],'c'): True
(because 'c' appears in the input list)

The mystery function takes as input a list and a value, and determines whether the value appears in the list.

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

b) Describe in English what the mystery function does.

10. Without looking at the posted code, write the function Koch(L, level) to draw the recursive Koch curve as shown below. (Recall that if level is 0, the turtle should simply go forward by L; in all other cases, the turtle should end up at the same position, distance L to the right of where it started.)



Level 0

Level 1

Level 2

Level 3

Level 4

```
def drawKoch(length, levels):
    if levels == 0:
        turtle.forward(length)
    else:
        drawKoch(length/3, levels-1)
        turtle.left(60)
        drawKoch(length/3, levels-1)
        turtle.right(120)
        drawKoch(length/3, levels-1)
        turtle.left(60)
        drawKoch(length/3, levels-1)
```