CSCI-141-01 Practice Midterm – SI Edition

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Disclaimer: These are examples of things which MIGHT appear on the midterm exam;

I make no guarantees that anything resembling them WILL appear. Rather than

memorizing the solutions to these select problems, use them to get an idea for

the processes you will be expected to know.

0. Introduction to Python

a. Why doesn't the following program run as expected?

def func():

print( 'CS is really cool!' )

print( 'I have had a really ' + \

input( 'Enter an adjective: ' ) + \

' time at RIT :n)' )

def foo():

print( 'I have had a really ', \

input( 'Enter an adjective: ' ), \

' time at RIT :n)' )

b. Assuming you fix the above problem, show what it would look

like when run in IDLE; include user input where necessary.

1. Conditional Execution and Functions with Parameters

a. Show the expected output of the following program:

def foo():

x = 10

print( 'x: ', x)

bar( x )

if x > 0:

print( 'x was unchanged' )

else:

print( 'x was changed' )

def bar( x ):

x -= 20

print( 'x: ', x )

foo()

b. Was this output expected? Why or why not?

2. General Recursion and Fruitful Functions

a. Draw an execution diagram for the following program:

from turtle import \*

def do\_stuff( param, other\_param ):

if( ( param > 0 ) and ( other\_param < 80 ) ):

fd( param )

lt( other\_param )

do\_stuff( param - 2, other\_param \* 2 )

do\_stuff( 100, 5 )

b. Show a substitution trace for the following program:

from math import \*

def do\_some\_other\_stuff( n ):

if( n == 0 ):

return 2

else:

return 2 \* sqrt( do\_some\_other\_stuff( n - 1 ) )

do\_some\_other\_stuff( 3 )

3. Tail Recursion, Loops, Assignment and Types

a. Convert the following code to be tail-recursive:

def rec( n ):

if( n == 0 ):

return 1

else:

return 3 \* rec( n - 1 )

b. Now make it iterative.

c. Locate and fix all errors in the following code (line numbers given

for reference) :

0 def some\_task():

1 input( 'Enter a number: n ' )

2 print( n )

3 n = input( 'Enter a new value for n: ' )

4 if( n < 10 ):

5 print( 'new value is less than 10' )

6 else:

7 print( "new value's greater than or equal to 10' )

4. Strings, Files, Complexity

You have received the following plain text file from an anonymous souce:

memo.txt

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dear wizards,

I write to inform you that

Vlad is a vampire. Please be

careful, as vampires are

dangerous.

Sincerely,

A Friend

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Your friend Vlad is not OK with this message, finding it libelious in

the extreme, and would like you to replace certain insulting words

within the document.

Each line of text is a separate line within the file. Write a function,

redact.py, which will scan through the document and replace every

instance of 'vampire' or 'vampires' with 'really cool guy'.

(As strings are immutable, simply print each newly editted line)

Assume that 'vampire' will appear at most once per line.

Do not use lists in your implementation.

Hint: How can we search through a line for a given word, using only

strings and what we know about them?

5. Searching and Sorting, Intro to Lists

a. Given a list, lst = [2,5,4,1,6], sort the list using Insertion Sort,

showing the contents of the list at each step.

b. Show a substitution trace of Binary Search called upon the above

list (after sorting) for the following values:

i. 4

ii. 1

iii. 3

c. What is the best-case runtime complexity of Insertion Sort? of

Binary Search? Why?

d. What is the worst-case runtime complexity of Insertion Sort? of

Binary Search? Why?

6. Greedy Algorithms, Intro to Classes

You want to go to the candy shop with your friends, but being a poor-

college kid, you are on a tight budget. You have also (for whatever

reason) been assigned as a project to keep a food journal.

a. Write class called Candy, which will represent an individual piece

of candy. Pieces of Candy have a Name, a Price (in cents), and a Sugar

Content (in grams).

b. Write a Maker function for the Candy Class from part a.

c. You think about the problem of how much candy you can buy given

how much money you have, and decide that you should base your

purchases off of a Greedy-Algorithmic approach. As the quality of candy

is directly proportional to its price (if price goes up, quality goes

up by a related amount), you decide that you are going to buy the most

expensive piece of candy you can afford, and if you have money left,

keep buying the most expensive piece you can afford (given your new

total at each step), until you can no longer afford any candy.

Write high-level pseudocode for this Greedy Algorithm. The algorithm

takes a list of types of candy, as well as an integer representing your

current money amount, and should return a list containing candy types

and how many of each type was purchased.

Hint: When do you know to stop the algorithm?

Hint: What do you do at each repeated step?

Hint: How should you represent the list returned by the algorithm?

Food For Thought: Consider how you would go about printing the results

of the algorithm, including what candy was purchased, in what quantities,

for what net price, and how many grams of sugar you just purchased.