**COMP 4030/6030** Spring 2018

Assignment 1

**Due:** Feb 6, 2018 (before class starts)

**Background**:

We discussed in class a technique called “decrease and conquer”. The main idea, abstractly, is that to solve a problem with API, say, f(A), we utilize the same strategy by invoking f(B), where the problem size of B is less than A.

Examples we discussed in class includes computing the length of a list:

def **rec\_len(L)**:

if L==[]:

return 0

else:

first = L.pop(0)

return **rec\_len(L)** + 1 # here’s the **decrease and conquer** step

Although the implementation of decrease and conquer can be iterative or recursive, a recursive implementation is often more natural and easier.

To do this assignment, please (a) review class materials for decrease and conquer, (b) make sure you know how to manipulate lists in Python, e.g. popping an item off the list and concatenating two lists.

**Assignment:**

1. (30 points) We discussed in class a decrease and conquer strategy to find the minimum number in a list. The Python function was like this:

def my\_min(L):

if L==[]:

print("undefined behavior.")

return

if len(L) == 1:

return L[0]

first = L.pop(0)

smallest\_of\_the\_tail = my\_min(L)

if first < smallest\_of\_the\_tail:

return first

else:

return smallest\_of\_the\_tail

Trace by listing in order all recursive function calls resulted from the following function call my\_min([10,5,20,7,10]).

1. (30 points) Employ the decrease and conquer strategy to write a recursive Python function to compute the sum of a list. Here’s the API:

# Input: a list of numbers

# Output: a number, which is the sum of all numbers in the list

def my\_sum( a\_list ):

# your code …

1. (30 points) Employ the decrease and conquer strategy to write a recursive Python function to reverse a list. Here’s the API:

# Input: a list of things

# Output: another list, which is the reverse of the input list

def my\_reverse( a\_list ):

# your code…

1. (10 points) Employ the decrease and conquer strategy to write a recursive Python function to compute the depth of a binary tree. The depth of a binary tree is the longest distance from the root to any leaf. Here’s the API:

# Input: root node of a binary tree

# Output: a number, which is the depth of the tree rooted by the input node

def my\_depth( a\_root\_node ):

# your code…

**Plagiarism Policy:**

You can discuss how to solve the problems with your classmates, but the code must be your own. Using other people’s code will result in a zero for the assignment and possible additional penalties.

**Submission**:

Put the solutions in one Python (.py) file. Put your name as part of the file name. For example: JohnSmith\_hw1.py.

Upload your **Python file** to the Assignment 1 folder in the eCourseware Dropbox.