"""

Thomas Morris

Insertion-sort program using a priority queue implemented with a sorted list.

November 8th, 2019

"""

from SortedPriorityQueue import SortedPriorityQueue

def insertion\_sort(C, S):

#add number from collection to sorted priority queue

j = 0

for i in C:

S.add(i, j)

j += 1

L = []

while not S.is\_empty():

L.append(S.remove\_min())

return L

if \_\_name\_\_ == '\_\_main\_\_':

#create a sorted priority queue

S = SortedPriorityQueue()

#create a collection of numbers

C = [7,4,8,2,5,3]

print(C)

C = insertion\_sort(C,S)

print(C)

"""

Thomas Morris

Implement a program which returns i) inorder and ii) preorder traversal of

a given a binary tree. Note that you should follow the code fragment style

to create your program below and can use recursive (or iterative) function.

November 8th, 2019

"""

#definion for a binary tree node

class TreeNode(object):

def \_\_init\_\_(self,x):

self.val = x

self.left = None

self.right = None

class Traversal(object):

def inorderTraversal(self,root):

#:Type root: TreeNode

#:rtype: List[int]

#left, root, right

#recursively traverse the tree

L = []

if root:

L = self.inorderTraversal(root.left)

L.append(root.val)

L = L + self.inorderTraversal(root.right)

return L

def preorderTraversal(self,root):

#:Type root: TreeNode

#:rtype: List[int]

#root, left, right

#recursively traverse the tree

L = []

if root:

L.append(root.val)

L = L + self.preorderTraversal(root.left)

L = L + self.preorderTraversal(root.right)

return L

if \_\_name\_\_ == '\_\_main\_\_':

N = TreeNode(1)

N.right = TreeNode(2)

N.right.left = TreeNode(3)

T = Traversal()

print(T.inorderTraversal(N))

print(T.preorderTraversal(N))