# Definition for a binary tree node.  
import math  
trunck = []  
class TreeNode(object):  
 def \_\_init\_\_(self, x):  
 self.val = x  
 self.left = None  
 self.right = None  
class YourSolution(object):  
 inOrderlist = []  
 preOrderlist = []  
 def inorderTraversal(self, root):  
 if root is None:  
 return  
 else:  
 left = self.inorderTraversal(root.left)  
 if left is not None:  
 self.inOrderlist.append(left)  
 now = root.val  
 if now is not None:  
 self.inOrderlist.append(now)  
 right = self.inorderTraversal(root.right)  
 if right is not None:  
 self.inOrderlist.append(right)  
 # :type root: TreeNode  
 # :rtype: List[int]  
 def preorderTraversal(self, root):  
 if root is None:  
 return  
 else:  
 now = root.val  
 if now is not None:  
 self.preOrderlist.append(now)  
 left = self.preorderTraversal(root.left)  
 if left is not None:  
 self.preOrderlist.append(left)  
 right = self.preorderTraversal(root.right)  
 if right is not None:  
 self.preOrderlist.append(right)  
#:rtype: List[int]  
#:type root: TreeNode  
print("Enter the branches of your tree")  
answer = int(input())  
trunck.append(TreeNode(answer))  
while answer != "no": # can put in many numbers into list  
 print("Enter the next branch, type none (if done input done")  
 answer = input()  
 if answer == "none":  
 answer = None  
 elif answer != "done":  
 answer = int(answer)  
 else:  
 break  
 trunck.append(TreeNode(answer))  
lim = math.ceil(len(trunck) - math.pow(2, (math.log2(len(trunck) + 1) - 1)))  
place = 0  
while place < lim:  
 # declare left and right  
 trunck[place].left = trunck[place + place + 1]  
 trunck[place].right = trunck[place + place + 2]  
 place -= -1  
test = YourSolution()  
test.inorderTraversal(trunck[0])  
test.preorderTraversal(trunck[0])  
print("in order: ", test.inOrderlist)  
print("Post order: ", test.preOrderlist)