1. (30 points)
Suppose that you are developing a PIM — Personal Information Management system which requires an ADT `addressTable` to operate a table of information cards. Each card represents a person or an organization, and contains data fields of *name* and *e-mail address*. Here, we assume that *name* is the key field and no cards have the same name.

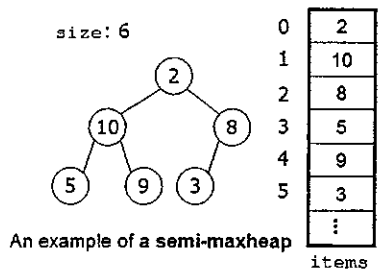
   (a) Please write a recursive method that inserts a pair of name and email address into a sorted (in ascending order, by *name*), pointer based implementation of the ADT.

   (b) Please write a recursive method that inserts a pair of name and email address into a pointer based binary search tree implementation of the ADT.

   (c) Beginning with an empty binary search tree, what binary search tree is formed when the following cards are inserted in the order given?
   <Jane, jane@gmail.com>,
   <Mary, mary@ntu.edu.tw>,
   <Allan, allan@acm.org>,
   <Lulu, lulu@hotmail.com>,
   <Steve, steve@yahoo.com>,
   <Nancy, nancy@gmail.com>,
   <Robert, robert@ntu.edu.tw>.

   (d) Please display the *names* of the above binary search tree in postorder manner.

2. (20 points)
A semi-maxheap is a complete binary tree whose left and right subtree are both maxheaps but the item in the root is out of place (see the following figure for an example).



An example of a **semi-maxheap**

(a) Please write a recursive function `void heapRebuild(int items[], int root, int size)` that converts a semi-maxheap rooted at index `root` into a maxheap. The parameter `items` represents an array of heap items (we assumed that the heap items are integers) and `size` indicates the number of items in the heap.

(b) Please write a function `void buildHeap(int items[], int size)` that uses the above `heapRebuild` function to transfer an array `items` of `size` integers into a maxheap. Given the array `int A[6] = {8, 11, 83, 3, 54, 77}`, show the content of the array after applying `buildHeap` to it.

3. (20 points)

Consider binary trees where each node stores a non-negative integer. Design an algorithm that, given such a tree T and a non-negative integer k as input, determines whether T contains a branch (from the root to a leaf) such that the sum of all numbers stored on the nodes of the branch equals k. Please present your algorithm in an adequate pseudo code and make assumptions wherever necessary. Give an analysis of its time complexity. The more efficient your algorithm is, the more points you will be credited for this problem. Is there a possibility that your code may overflow? Have you avoided the problem?

4. (30 points)

Finding the longest path of an arbitrary graph is a hard problem. However, efficient algorithms exist for certain types of graphs such as *acyclic directed graphs* (which are graphs without directed cycles). Design an efficient algorithm to compute the length of the longest path of a given acyclic directed graph $G = (V, E)$. Assume that G is given by adjacency lists. Please present your algorithm in an adequate pseudo code and make other assumptions wherever necessary. Explain why your algorithm is correct and give an analysis of its time complexity. The more efficient your algorithm is, the more points you will be credited for this problem.