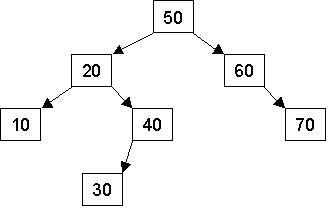
**Homework 5**

**Time due: 9:00 PM Thursday, June 7**

This homework is a good study guide for the final. The final will be open book, open notes. We don't expect you to memorize every last detail of every algorithm, so a skill you should develop is the ability to locate an algorithm in a reference source, trace through it, and understand it. If it's not in exactly the form your application requires, you should be able to adapt it.

1. Consider the following binary search tree, ordered using the < relationship:



* 1. Using the simplest binary search tree (BST) insertion algorithm (no balancing), show the tree that results after inserting into the above tree the nodes 80, 65, 75, 45, 35 and 25 in that order. (If you're not skilled with a drawing tool, use a simple text form of the tree. For example, the tree depicted above could be shown as
  2. 50
  3. 20 60
  4. 10 40 70
  5. 30

Use enough space to distinguish left children from right children. Another way to represent the tree in text form (that distinguishes left children from right children) is

50

20

10

40

30

xx

60

xx

70

* 1. After inserting the nodes mentioned in part a, what is the resulting BST after you delete the node 30, then the node 20? (Again, just use a simple deletion algorithm with no balancing.)
  2. After inserting the nodes mentioned in part a, what would be printed out by in-order, pre-order, and post-order traversals of the tree (assume your traversal function prints out the number at each node as it is visited)?

1. (We won't not get to the material you need to answer this until Monday, so unless you look at Chapter 12 in the book or watch the [online heaps lecture](http://www.cs.ucla.edu/classes/spring12/cs32/Codeexamples/index.html), you'll probably hold off on this problem until after Monday's class. It's not one that takes much time to do.)

Consider the following operations on an initially empty heap h ordered by the < relationship. (This heap is a maxheap: the biggest item is at the top). The heap is represented as a binary tree:

h.insert(3);

h.insert(5);

h.insert(2);

h.insert(1);

h.insert(9);

h.insert(4);

int item;

h.remove(item); // Removes the biggest item from the heap, and puts it in item

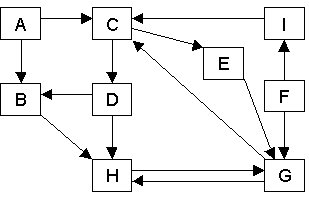
h.insert(8);

h.insert(7);

h.remove(item);

* 1. Show the resulting heap (As in problem 1a, show the tree in some recognizable form.)
  2. Show how your heap from part a would be represented in an array.
  3. Remove the top item from the heap and show the resulting array after the removal operation.

1. In some binary search trees, each node has a left child pointer, a right child pointer and a parent pointer. The parent pointer of a node points to its parent (duh!), or is NULL if the node is the root node. This problem will examine such trees.
   1. Show a C++ structure/class definition for a binary tree node that has both child node pointers and a parent node pointer. Assume the data stored in each node is an int.
   2. Write pseudocode to insert a new node into a binary search tree with parent pointers. (Hint: You can find binary search tree insertion code on pp. 543-545 & 560-561 of the course textbook).
2. (We might not get to the material you need to answer this until Wednesday, so unless you look at Chapter 13 in the book, you'll probably hold off on this problem until after Wednesday's class. It's not one that takes much time to do.)
   1. Show an adjacency matrix and an adjacency list for the following graph.



* 1. If you perform a depth-first traversal through this graph starting from vertex E, what vertices are visited, and in what order? There is more than one answer to this question, so list all the answers.

**Turn it in**

By Wednesday, June 6, there will be a link on the class webpage that will enable you to turn in this homework. Turn in one zip file that contains your solutions to the homework problems. The zip file should contain one file:

* hw.doc, hw.docx, or hw.txt, a Word document or a text file with your solutions to the problems.