Homework Set #9

1. Exercise 12.1-2 (page 289, 10 points)

   In a heap, a node's key is >= both of its children's keys. In a binary search tree, a node's key is >= its left child's key, but <= its right child's key.

   The heap property, unlike the binary-searth-tree property, doesn't help print the nodes in sorted order because it doesn't tell which subtree of a node contains the element to print before that node. In a heap, the largest element smaller than the node could be in either subtree.

   Note that if the heap property could be used to print the keys in sorted order in O(n) time, we would have an O(n)-time algorithm for sorting, because building the heap takes only O(n) time. But we know (Chapter 8) that a comparison sort must take Ω(n lg n) time.

2. Exercise 12.2-5 (page 293, 10 points)

   Let x be a node with two children. In an inorder tree walk, the nodes in x's left subtree immediately precede x and the nodes in x's right subtree immediately follow x. Thus, x's predecessor is in its left subtree, and its successor is in its right subtree.

   Let s be x's successor. Then s cannot have a left child, for a left child of s would come between x and s in the inorder walk. (It's after x because it's in x's right subtree, and it's before s because it's in s's left subtree.) If any node were to come between x and s in an inorder walk, then s would not be x's successor, as we had supposed.

   Symmetrically, x's predecessor has no right child.

3. Problem 12-1 (page 303, 40 points)

   a. O(n^2). After we insert the first item, we insert all other items as a right leaf of the tree. Each time we insert an item into the tree, the tree height increases by 1. When we insert the k-th item, we run the while loop (begin at line 3) k-1 times. Thus, to insert n identical items, the total time is $\sum_{i=1}^{n} i$ = O(n^2).

   b. O(n lg n)

      With this strategy, we will build and maintain a balanced tree. The tree height $h = \lfloor lg\ k \rfloor$, where k is the number of items in the tree. Since the procedure TREE-INSERT runs in O(h) time, to insert n identical items the total time cost is O(lg n).

   c. O(n), because each time we insert an item to the head of the list.

d. Worst case running time $O(n^2)$ – all items insert into one side, it will be same as (a).

   Expected running time $O(n \lg n)$ – the binary tree is a balanced tree, it will be same as (b)