Score: / 41 CIS 263

Name: last, first 2011:Nov:02(Wed)

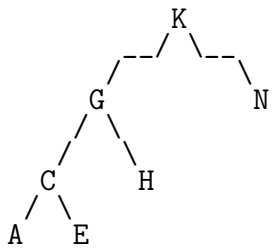
Midterm Examination

- During this test, you should not use any auxiliary materials or computational devices; i.e. this test is 'closed-book' (and 'closed-notes' etc.). But if you can't remember some little detail, ask the instructor.

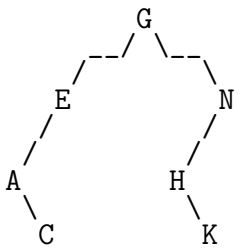
[Acknowledgment: Some of these exercises are derived from our textbook by Weiss.]

A. Consider the two different ordered binary search trees provided here, tree #1 and tree #2:

tree #1:



tree #2:



tree #3:

1. [1 point] Which of the two provided trees has the inorder traversal `A,C,E,G,H,K,N`? Circle your answer(s): tree #1 tree #2
2. [1 point] Which of the two provided trees has the preorder traversal `K,G,C,A,E,H,N`? Circle your answer(s): tree #1 tree #2
3. [1 point] Which of the two provided trees has the postorder traversal `A,E,C,H,G,N,K`? Circle your answer(s): tree #1 tree #2
4. [2 points] In the space at right, draw the ordered binary search tree that will have the preorder traversal `K,H,N`:
5. [1 point] Give the inorder traversal of the tree that you drew immediately above:
6. [3 points] In the space at right, draw the ordered binary search tree that will have the postorder traversal `C,G,H,E`:
7. [1 point] Give the inorder traversal of the tree that you drew immediately above:
8. [7 points] In the space below the caption "tree #3" at the beginning of this sequence of exercises, draw the ordered binary search tree that will have the preorder traversal `E,C,A,H,G,K,N`
9. [1 point] Give the inorder traversal of the tree #3 that you drew:
10. [3 points] Adjacent to every (non-`NULL`) node in tree #1 above, write its height.

Here is our code for `insert()` for unbalanced ordered binary search trees:

```
void insert( const Comparable & x ) { insert(x, root); }
void insert( const Comparable & x, BinaryNode * & t ) {
    if ( t == NULL )
        t = new BinaryNode(x, NULL, NULL);
    else if ( x < t->element )
        insert(x, t->left);
    else if ( t->element < x )
        insert(x, t->right);
    // debugging/developing:
    cout << t << endl;
    }
```

11. [2 points] If the value `D` will be added in tree #1 above using the code at the top of this page, then what will be the (debugging/development) output from this code? (Assume that outputting nodes will output labels used above, e.g. "`K`", "`A`", "`N`", etc.)

12. [10 points] Suppose `struct BinaryNode` has a publicly accessible member variable (a.k.a. field) `int height`. Rewrite the second method `insert()` at the top of this page so it will also efficiently set/reset `t->height` as appropriate. (In case you might wonder, a practical use for such code is for the AVL scheme for keeping ordered binary search trees balanced.) If you want, you can simply write code to be added at the end of the provided code. Be sure to avoid trying to deference a `NULL` pointer. If you want to use an additional method or function, you need to write it here. (Except you may assume that the basic function `max(a,b)` is provided for you, if you want to use it.)

B. [8 points] Write a C++ function implementing binary search to return the index of a target value in an array of `doubles`, or return -1 if the target value is not in the array. Arguments for this function will be the desired target value, the array, and two parameters L and R specifying the indexes of the leftmost and rightmost elements (inclusive). You should assume that the array is already in order; you do **not** need to sort it. For example, `binary_search(105.04, {-300.1, -13.12, -10.0, 84.4, 94.4, 105.04, 123.12}, 0, 6)` should return 5.
You can do this work recursively or iteratively.

```
int
binary_search(double target, double material[], int L, int R)
{




}
```