CSI2114 - Spring 2006 - Assignment 3

Due: June 23, 2006 by 5 pm (in Box # 5)

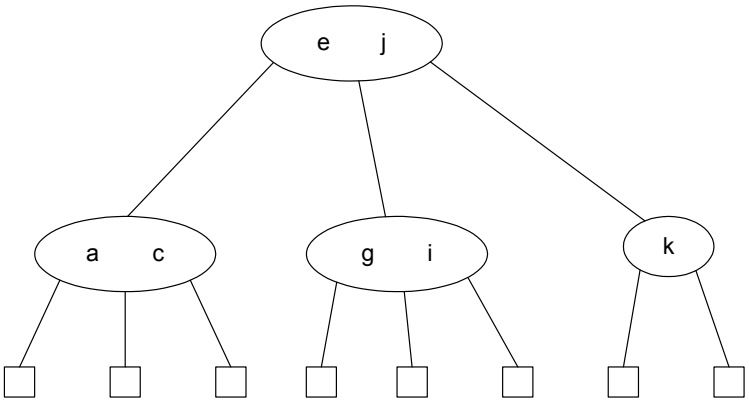
Exercise 1. [20 points]

a) Let T be a binary search tree, and let x be a key. Give an efficient algorithm for finding the smallest key y in T such that $y > x$. Note that x may or may not be in T. Explain why your algorithm has the running time it does.

b) Write a non-recursive algorithm to print out the keys from a binary search tree in order.

Exercise 2. [20 points]

a) Starting with the 2-4 tree shown below, perform the following operations (in order): insert *h*, insert *f*, insert *b*, insert *d*, delete *d*, delete *j*. Show the tree after each operation, as well as after any intermediate split/swap/transfer/fusion steps performed.



b) Starting with an empty 2-4 tree, what is the minimum number of keys one would have to insert so that inserting the last key causes multiple splits? Give an example of such a sequence of keys. Draw the tree both before the insertion of the final key and after that insertion (your example should not have more than 15 keys).

Exercise 3. [10 points]

Draw the 11-item hash table that results from using the hash function *h(i) = (2i + 5) mod 11* to hash the keys *12, 44, 13, 88, 23, 94, 11, 39, 20, 16*, and *5*, assuming collisions are handled by double hashing. Use *h'(i) = 7 – (i mod 7)* for the secondary hash function. In both hash functions, the argument *i* is the value of the key being hashed.

Exercise 4. [30 points]

Write a JAVA program to implement a data-structure called "Country_Dictionary", which will store the names of countries of the world, using the first letter of the name as the key. For example, the countries "Angola", "Argentina", "Australia" would be stored under the key 'A'. The data structure "Country-Dictionary" should have the following operations:

insert(String CountryName) : Insert the country name that is given (should return error if it already exists)
delete(String CountryName) : Delete the country name that is given (other countries with same key should not be deleted)
search(char key) : Print all countries that start with the character given as key.

To implement this data-structure, you will use a binary search tree. Each node of the tree should store a key (i.e. a char) and a list of countries under that key. To insert a country name, extract its first letter (which will be the key) and search for this key; If there is node in the tree with that key, you should add the new country name in the list at this node (unless it already exists there). Otherwise, you will create a new node with that key and add the country name to the list of this new node. Similarly, when deleting a country name, you should remove it from the corresponding list. You should delete a node from the tree only when the list for that node becomes empty.

The main program should look as follows:

```
public static void main() {
        Country_Dictionary Cdict = new Country_Dictionary();
        Cdict.insert("Canada"); Cdict.insert("Australia");Cdict.insert("Germany");
        Cdict.insert("USA");Cdict.insert("Japan");Cdict.insert("UK");Cdict.insert("Argen
tina");
        Cdict.insert("Mexico");Cdict.insert("Belgium");
        Cdict.search('U');
        Cdict.search('G');
        Cdict.insert("Angola");
        Cdict.delete("Germany");Cdict.delete("Canada");
        Cdict.insert("Chile");
        Cdict.search('G');
        Cdict.search('C');
        Cdict.search('A');
}
```

Test your program before you submit it. You may use any of the methods or classes from the textbook, but make sure that it works.