Data Structure and Algorithm
Homework #3

Due: 2:00pm, Tuesday, April 10, 2012

TA email: dsa1@csie.ntu.edu.tw

=== Homework submission instructions ===

- For Problem 1, submit your source code, a shell script to compile the source, and a brief documentation to the SVN server (katrina.csie.ntu.edu.tw). You should create a new folder "hw3" and put these three files in it.
- The filenames of the source code, the shell script, and the documentation file should be "dictionary.c", "compile.sh", and "report.txt", respectively. **The shell script should compile your source codes to generate an executable binary with the filename 'dictionary'.** You will get some penalties in your grade if your submission do not follow the naming rule.
- The documentation file should be in plain text format (.txt file). In the documentation file you should explain how your code works, and anything you would like to convey to the TAs.
- For Problem 2 to Problem 5, submit the answers through the CEIBA system (electronic copy) or to the TA in R508 (hard copy).
- Each student may only choose to submit the homework in one way; either all as hard copies or all through CEIBA except the programming assignment. If you submit your homework partially in one way and partially in the other way, you might only get the score of the part submitted as hard copies or the part submitted through CEIBA (the part that the TA chooses).
- If you choose to submit the answers of the writing problems through CEIBA, please combine the answers of all writing problems into only ONE file in the doc/docx or pdf format; otherwise, you might only get the score of one of the files (the one that the TA chooses).
- Discussions with others are encouraged. However, you should write down your solutions by your own words. In addition, you have to cite the sources you consulted from or people you discussed with on the first page of your solution to that problem. The TA can deduct up to 100% of the score for any problem without the citation.
- No late submission of the homework will be given any score (for that portion).

***Problem* 1.** Binary Search Tree (40%)

In this problem, we asked you to use the binary search tree to construct a dictionary. You should start with an empty binary search tree. A list of N words will be given to you to construct the initial binary search tree. They should be inserted into the binary search tree in the same order specified in the list. You should use the following assumptions/definitions:

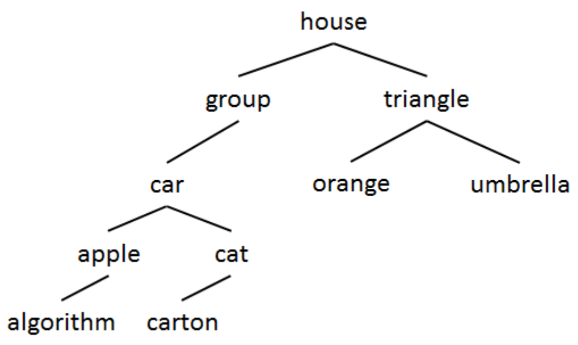
1. Following the same rules as a regular binary search tree, for each node associated with word X, all words associated with nodes in the left sub-tree should always be smaller than X **alphabetically**, and all words associated with nodes in the right sub-tree should always be larger than X **alphabetically** (see the next item for the definition).

2. Each word A consists of L_a letters $\{a_1a_2a_3 \cdots a_{L_a}\}$, where $a_i, 1 \le i \le L_a$, are the ASCII code of each letter in the word. If word A and word B are different words, we say that word A is larger than word B alphabetically if $j = \operatorname*{argmin}_{1 \le i \le \min(L_a, L_b)} (a_i \neq b_i)$ and $a_j > b_j$. Similarly, we say that word A is smaller than word B alphabetically if $j = \operatorname*{argmin}_{1 \le i \le \min(L_a, L_b)} (a_i \neq b_i)$ and $a_j < b_j$. In the case that $a_i = b_i, \forall 1 \le i \le \min(L_a, L_b)$ and $L_a \neq L_b$, then word A is larger than word B alphabetically if $L_a > L_b$ and smaller than word B alphabetically if $L_a < L_b$ (e.g., "note" is alphabetically smaller than "notebook"). Finally, word A equals to word B alphabetically if they are the same word.

3. The input words always consist of lowercase letters (from 'a' to 'z'), and each word contains at most 20 letters.

4. The input words are all distinct.

For example, if the given list is {"house", "group", "triangle", "orange", "car", "cat", "apple", "algorithm", "carton", "umbrella"}, the initial binary search tree will be as follows:



After building the initial binary search tree, your program needs to support following operations:

1. **INSERT**: Insert a new word into the binary search tree. If the word to be inserted already exists in the tree, output "error".

2. **DELETE**: Delete the specified word from the binary search tree. If the word does not exist in the tree, print "error". If the node with the word to be deleted in the tree has two children,

the node should be replaced by the node with the largest word in its left sub-tree. If the node has only one child, the node should be replaced by its child. If the node is a leaf, just delete the node and return NULL to its parent.

3. **INORDER**: Traverse your tree in inorder.
4. **PREORDER**: Traverse your tree in preorder.
5. **POSTORDER**: Traverse your tree in postorder.

- Input Format

  N $\leftarrow$ the number of the words for build a binary search tree ($N \leq 1,000,000$)

  word1 word2 word3 ... wordN $\leftarrow$ A list of N words to construct the initial binary search tree

  INSERT word $\leftarrow$ the operation and the word to be used in the operation

  DELETE word $\leftarrow$ the operation and the word to be used in the operation

  INORDER $\leftarrow$ the operation

  PREORDER $\leftarrow$ the operation

  POSTORDER $\leftarrow$ the operation

  $\vdots$

  (the total number of operations $\leq 1,000,000$)

- Sample Input

```
10
house group triangle orange car cat apple algorithm carton umbrella
PREORDER
INORDER
POSTORDER
INSERT triangle
DELETE car
INORDER
INSERT hawk
PREORDER
DELETE teacher
```

- Sample Output

```
house group car apple algorithm cat carton triangle orange umbrella
algorithm apple car carton cat group house orange triangle umbrella
algorithm apple carton cat car group orange umbrella triangle house
```

```
error
algorithm apple carton cat group house orange triangle umbrella
house group apple algorithm cat carton hawk triangle orange umbrella
error
```

Problem 2. Left Child Right Sibling Representation (15%)

Suppose we use the following declaration for a program to represent a node in a degree-3 tree.

```
struct nodeTree {
  SomeType data;
  struct nodeTree *child1;
  struct nodeTree *child2;
  struct nodeTree *child3;
};
typedef struct nodeTree NodeTree;
```

1. (5%) Design an algorithm to convert a degree-3 tree into a binary tree with the LC-RS representation (referred as the LC-RS tree). The input of the algorithm is a pointer which points to the root node in the degree-3 tree. You can assume that in every node of the degree-3 tree, the pointers to child nodes are first assigned to child1, then to child2 if there is a second child, and then to child3 if there is a third child. That is, if a node only has 2 child nodes, only child1 and child2 will have valid pointers, and child3 would equal to NULL. The output of the algorithm should be a pointer that points to the root node in the LC-RS tree. You should utilize the following structure to represent a node in the LC-RS tree.

   ```
   struct nodeLCRS {
     SomeType data;
     struct nodeLCRS *left;
     struct nodeLCRS *right;
   };
   typedef struct nodeLCRS NodeLCRS;
   ```

2. (5%) Design an algorithm to print all the data in a LC-RS tree. The algorithm should run in $O(n)$ time, where n is the number of nodes in the LC-RS tree.

3. (3%) Which traversal method does your algorithm in (2) use? Preorder? Inorder? Postorder? Please explain.

4. (2%) Utilize your algorithm in (2) to traverse the LC-RS tree in p.13 of the slide deck “tree 1”. Please show the results.

Problem 3. Binary Tree and Binary Search Tree (20%)

1. (6%) Given a binary search tree T and a pair of numbers of (a, b) where $a \leq b$, give an algorithm that returns the set of elements in T with key values in $[a, b]$. Your algorithm should run in $O(h + m)$ -time, where h is the height of T and m is the number of elements within the range.

2. (4%) Prove the following statements:

   (a) If a node in a Binary Search Tree has two children, then its predecessor in the inorder traversal has no right child.

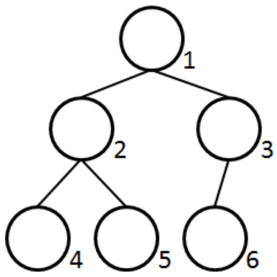
   (b) If a node in a Binary Search Tree has two children, then its successor in the inorder traversal has no left child.

3. (5%) Insert the following integers into an initially empty binary search tree in the given order, and draw the resulting tree: 8, 2, 9, 1, 5, 6, 3, 7.

4. (5%) Draw the binary tree whose in-order and post-order traversals of the nodes are:
   In-order: G D P K E N F A T L
   Post-order: G P D K F N T A L E

Problem 4. Max Heap (15%)

We can use an array to represent a max heap. For example, to represent the following heap,



, you can use the number at the bottom-right of each node as the index of the array. (You can assume that the index of each array begins with 1.)

1. (7%) An array $A[]$ is "heapified" *iff* $A[i] \leq A[\lfloor i/2 \rfloor], \forall\, 2 \leq i \leq n$ (n is the number of elements in A). Given an array which contains n elements and is not "heapified", design an algorithm to transform the given array to become "heapified" according to the definition. In addition, you should calculate the time complexity of your algorithm using the Θ notation.

2. (4%) Let array $A[]$ be a max heap and $A[1] = 50, A[2] = 25$. Please list the elements of array $A[]$ after inserting 66, 33, 34, and 75 (in the given order). You should describe how array $A[]$ is modified in a step-by-step manner.

3. (4%) Continue with problem 4.2, please list the elements of array A[] after deleting 33 and 66. To delete one element in the heap, the last element in A[] will replace the deleted element and elements in A[] will be re-organized so that it still satisfies the "heapified" condition. You should describe the change of array $A[]$ in a step-by-step manner.

Problem 5. Threaded Binary Tree (16%)

In order to represent a tree structure, linked structure is often utilized. The number of pointers of in each tree node depends on the degree of the tree. However, some pointers in the node structure equal to NULL and are wasted.

To prevent this problem, one idea is to store some useful information in these NULL pointers. The inorder threaded binary tree stores the inorder traversal information in the originally NULL pointers. If the node has no left child, its left-child pointer will point to the node's inorder predecessor. If the node has no right child, its right-child pointer will point to the node's inorder successor. Two additional variables will be added to the node structure to record whether each pointer is a link (points to a child) or a thread (points to its inorder predecessor or successor).

1. (6%) According to the above definition, describe which nodes will be pointed by the left-thread and right-thread pointers of a newly inserted **leaf node** for following two situations:

   (a) If the new node is the left-child of its parent.

   (b) If the new node is the right-child of its parent.

   Write down your answer, and briefly explain the reason.

2. (10%) Given an inorder threaded binary tree, please design an algorithm to find the parent of a given node. Write down your algorithm using C or pseudo-code. Note that you have to start the traversal with the given node.