/* 是负号有两面 */

Data Structures
Spring 2009, Midterm Exam II, May 7
7:00-9:00 PM

1. (30%) Give answers to the following short questions.

   (a) (5%) Draw the binary tree if the inorder sequence is [d, e, b, a, f, h, c, g] and the postorder sequence is [e, d, b, h, f, g, c, a].

   (b) (5%) Use an example to demonstrate that we cannot uniquely determine a binary tree based on its preorder and postorder sequences. (That is, you need to construct two different binary trees that have exactly the same set of preorder and postorder sequences.)

   (c) (5%) Suppose that each node in a given binary tree either has two children or has no child at all. If the number of leave nodes (terminal nodes) is 10, what is the number of internal nodes (non-terminal nodes)?

   (d) (5%) How many null links do we have in an n-node binary tree of linked representation?

   (e) (5%) For non-recursive versions of preorder, inorder, postorder, which auxiliary data structure is needed ? Which traversal method needs the use of a queue?

   (f) (5%) To check if a Binary tree is a Binary Search Tree, we can utilize tree traversal algorithm. Which traversal can be used ? If the tree is indeed a BST, what traversal result will we obtain ?

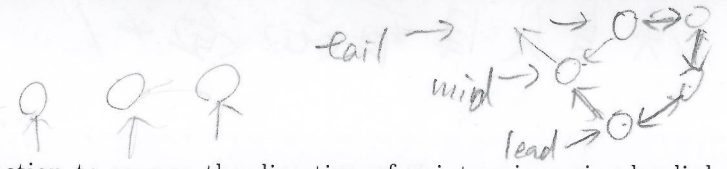
2. (10%) Suppose that we use an array of integers to implement a min-heap. Please show the contents of the array after each of the following operations is performed (starting with an empty heap): insert 47, insert 35, insert 78, insert 14, delete min, insert 51, insert 3, delete min.

3. (10%) Draw the binary search tree during the following operations, beginning with an empty tree: insert 47, insert 35, insert 78, insert 14, insert 16, insert 15, draw the tree, delete min, draw the tree, insert 60, insert 80, insert 79, delete max, draw the tree, delete 47, draw the tree. (You only need to draw the tree 4 times in the answer sheet, as indicated in the operation sequence. Moreover, if there are multiple possible trees after some of the operations, you need to draw all of them.)

4. (10%)

   (a) (5%) What is the weighting rule of union operation in set data structure.

   (b) (5%) Prove that a tree T with m nodes created by a sequence of unions using weighting rule, the height of T is no more than $\lfloor log_2 m \rfloor + 1$.



5. (10%) Write a function to reverse the direction of pointers in a circular linked list. (Remember to give the data type definition first.)

6. (10%) Write a RECURSIVE C (or C-like) procedure *treepointer copy* (*treepointer orig*) that copies a binary trees pointed by *orig* and returns the copied tree. Let the tree node be defined as :

```
typedef struct treenode *treepointer;
typefef struct treenode {
  int key;
  treepointer lchild, rchild;
} NODE;
```

7. (10%) A weighted tree is a Binary tree in which each node ni is associated with a weight, $weight(ni)$. The weight of a path from a root to a node is defined as the summation of weights of all nodes along the path. The largest weighted path is the path which has the largest weight among all root-to-leave (terminal node) paths. Write a C (or C-like) recursive procedure *int LargestWeightedPath*($NODE$ $*$ *root*) that returns the largest weighted path of the tree. Let the tree node be defined as :

```
typedef struct treenode {
  int weight;
  struct treenode *lchild; /* Pointer to the left subtree */
  struct treenode *rchild; /* Pointer to the right subtree */
} NODE ;
```

8. (10%) Let nodes of a Binary Search Tree be defined as

```
typedef struct treenode *treepointer;
typedef struct treenode {
  int key;
  int rightsize;
  treepointer lchild, rchild;
} NODE ;
```

where *rightsize* is the number of elements in the right subtree. Write a procedure *treepointer search* (*treepointer t*, *int k*, *int n*) which search the *kth* smallest element, where n is the number of nodes in the tree pointed by t.