Department of Computer Science and Engineering
National Sun Yat-sen University
Data Structures Quiz, Chapter 5, Dec. 7, 2015

1. Extend the array representation of a *complete binary tree* to the case of *complete trees* whose degree is *d*, *d*> 1. Develop the formulas for the parent and children of the node stored in position *i* of the array. Here, the root of the tree is stored in position 1 of the array. (20%)

2. Write a recursive C++ function to count the number of leaf nodes in a binary tree. (40%)

```
class TreeNode {
   int data;
   TreeNode *leftChild, *rightChild;
};
int count( TreeNode *root)
// Return number of leaf nodes in the binary tree pointed by "root".
// Return 0 if the binary tree is empty.
{
```

Please write the body of count ( ).

```
} // end of count ( )
```

3. Write a C++ function to insert a new node *r* as the right child of node s in a threaded binary tree. The right subtree of *s* becomes the right subtree of *r*. (40%)

```
class TreeNode {
   int data;
   TreeNode *leftChild, *rightChild;
   bool leftThread,rightThread; //leftThread, rightThread
     /* if rightThread == true, then rightChild is a thread
              (pointer to inorder successor)
        otherwise, rightChild is a pointer to the real right child */
};
void insertR(TreeNode *s, TreeNode *r)
{
```

Please write the remaining body of insertR ( ).

```
   if (! r -> rightThread) {// rightChild is not a thread
      treeNode *q = InorderSucc (r); // return inorder successor of r
      q -> leftChild = r;
   }
} // end of insertR ( )
```

Answer:

1. parent of i: $\left\lfloor \frac{i+(d-2)}{d} \right\rfloor$

children of i: di-d+2, di-d+3, di-d+4, …, di, di+1

2.

```
int count( TreeNode *root)
{
      if(root == 0)
            return 0;
      if(root->leftChild == 0 && root->rightChild == 0)   //必須寫以下這兩行
            return 1;
      else
            return count(root->leftChild) + count(root->rightChild)
}
```

3.

```
void InsertR(treeNode *s, treeNode *r)
{// Insert r as the right child of s.
   r -> rightChild = s -> rightChild;
   r -> rightThread = s -> rightThread;
   r -> leftChild = s;
   r -> leftThread = True; // leftChild    is a thread
   s -> rightChild = r;
   s -> rightThread = false;
   if (! r -> rightThread) {// rightChild is not a thread
      ThreadedNode <T> *q = InorderSucc (r);
      // return the inorder successor of r
      q -> leftChild = r;
   }
}
```