Department of Computer Science and Engineering
National Sun Yat-sen University
Data Structures Quiz, Chapters 5, Nov. 11, 2024

1. Let b_n denote the number of distinct binary trees with n nodes. Please present the recurrence formula for computing b_n. (20%)

2. Write a recursive C++ function to return the maximum value of the nodes stored in a binary tree (not a binary search tree), where each node stores one positive value in "data". (40%)

```
class TreeNode {
   int data;
   TreeNode *left, *right;    // pointer to leftChild and rightChild
};
int maxB( TreeNode *root)
// Return maximum value of the binary tree pointed by "root".
// Return -9999 if the binary tree is empty.
{
```

Please write the body of maxB ( ).

```
} // end of maxB ( )
```

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* after the insertion. (40%)

```
class TreeNode {
   int data;
   TreeNode *leftChild, *rightChild;
   bool 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)
{
```

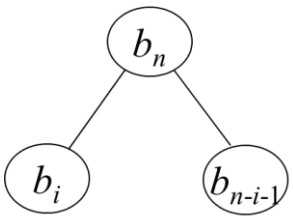
Please write the remaining body of insertR( ). // Insert r as the right child of s.

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

Answers:

1.

$$b_n = \sum_{i=0}^{n-1} b_i b_{n-i-1}, \quad n \geq 1, \text{ and } b_0 = 1, b_1 = 1$$



2.

```
class TreeNode {
   int data;
   TreeNode *left, *right;
};
int maxB( TreeNode *root)
// Return maximum value of the binary tree pointed by "root".
// Return -9999 if the binary tree is empty.
{
int leftM, rightM;
if( root == 0 )
   return -9999;      // Return -9999 if the binary tree is empty.
leftM = maxB( root->left );
rightM = maxB( root->right );
if ( leftM >= rightM )
   return root->data > leftM ? root->data : leftM;
            // left subtree is larger
else
   return root->data > rightM ? root->data : rightM;
            // right subtree is larger
} // end of maxB ( )
```

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
       TreeNode *q = InorderSucc (r); // return the inorder successor of r
       q -> leftChild = r;
    }
}
```