Finding the Inorder Successor of Node

```
threadedPointer insucc(threadedPointer tree) {
/* find the inorder sucessor of tree in a threaded
   binary tree */
    threadedPointer temp;
    temp = tree->rightChild;
    if (!tree->rightThread) // rightChild exists!
        while (!temp->leftThread)
            temp = temp->leftChild;
    return temp;
}
```

To perform an inorder traversal, we can simply make repeated calls to insucc!

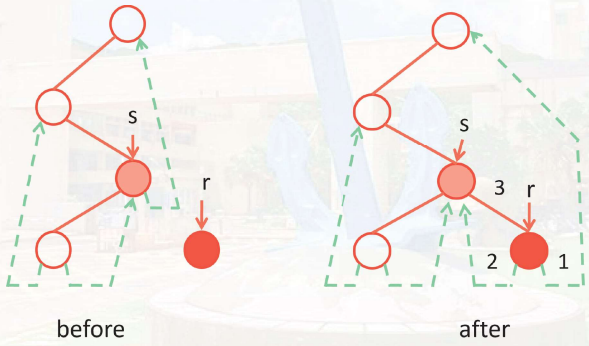
Inorder Traversal of a Threaded Binary Tree

```
void traverseInorder(threadedPointer tree) {
/* traverse the threaded binary tree inorder */
    threadedPointer temp = tree;
    while (1) {
        temp = insucc(temp);
        if (temp == tree)
            break;
        printf("%3c", temp->data);
    }
}
```

- **Note:** `temp == tree` happens when the last node is visited (then the successor becomes the header node).

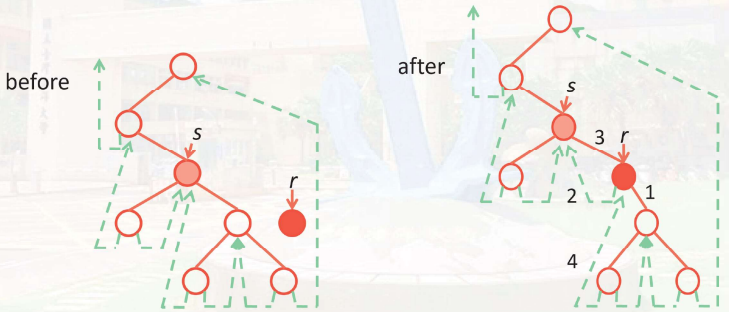
Inserting *r* as the rightChild of a node *s*

- Case I: `s->rightThread == true` (*s* has an empty subtree)



Inserting *r* as the rightChild of a node *s*

- Case II: `s->rightThread == false`
  (the right subtree of *s* is not empty)



The Code for the Insertion

```
void insertRight (threadedPointer s,
                  threadedPointer r) {
/* insert r as the right child of s */
    threadedPointer temp;
    r->rightChild = s->rightChild;
    r->rightThread = s->rightThread; // (*)
    r->leftChild = s;
    r->leftThread = true;
    s->rightChild = r;
    s->rightThread = false;
    if (!r->rightThread){ // step 4 (*)
        temp = insucc(r);
        temp->leftChild = r;
    }
}
```