CS 241 Data Organization More List and Tree Fun

Brooke Chenoweth

University of New Mexico

Fall 2014

Pointer Changing Code

What do we do if a function needs to change one of the pointer parameters passed to it?
Assume we have `struct Node* root` that points to a tree.

- We could use pointers to pointers.

```
void changeTree(struct Node** node);
```

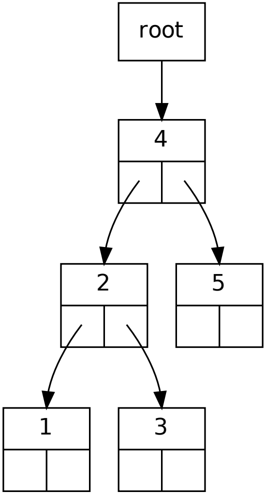
```
changeTree(&root);
```

- We could have the function return new pointer value.

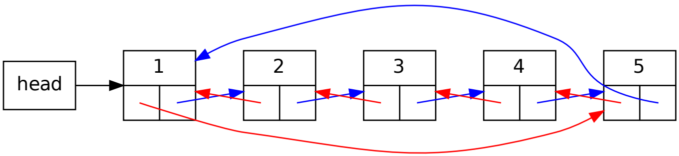
```
struct Node* changeTree(struct Node* node);
```

```
root = changeTree(root);
```

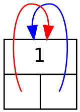
Ordered Binary Tree



Circular Doubly Linked List



First and last nodes wrap around to each other.
Null pointer represents an empty list.



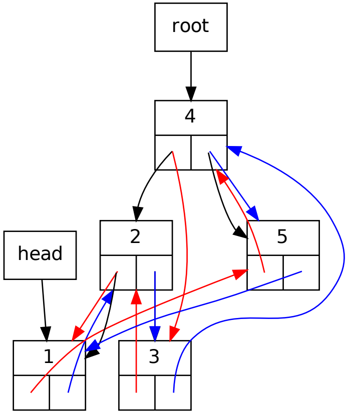
Length 1 list looks a bit silly. . .

Common Node “Shape”

```
struct Node
{
  int data;
  struct Node* left;
  struct Node* right;
};
```

- In tree, right and left are greater and lesser subtrees.
- In list, right and left are next and previous nodes.

Tree to List Challenge



Take ordered binary tree and rearrange the pointers to make a circular doubly linked list.

This operation can be done in O(n) time.

Tree to List

```
/* Given root node of binary tree,
 * convert to list and return head node.
 */
struct Node* treeToList(struct Node* node);
```

- How will this function work?
- What helper function(s) will we want?

List helpers

```
/* Given two circular doubly linked lists,
 * append them and return new head node.
 */
struct Node* joinLists(struct Node* a,
                       struct Node* b);

/* Join two nodes so second follows first. */
void joinNodes(struct Node* a, struct Node* b)
{
  a->right = b;
  b->left = a;
}
```

joinLists

```c
struct Node* joinLists(struct Node* a,
                       struct Node* b)
{
  struct Node* aLast;
  struct Node* bLast;

  if(a == NULL) return b;
  else if(b == NULL) return a;
  else
  {
    aLast = a->left;
    bLast = b->left;

    joinNodes(aLast, b);
    joinNodes(bLast, a);
  }
  return a;
}
```

treeToList

```
struct Node* treeToList(struct Node* node)
{
  struct Node* left;
  struct Node* right;

  if(node != NULL)
  {
    left = treeToList(node->left);
    right = treeToList(node->right);

    node->left = node;
    node->right = node;

    node = joinLists(left, node);
    node = joinLists(node, right);
  }
  return node;
}
```