Binary Trees

0, 1, or 2 children only
Need linked list siblings?

• NO!
• Why not? Because this time we know we have only 1 or 2 children.
Binary Node Implementation

In Java:

```java
public class BinaryNode {
    int value; //the data in the node
    BinaryNode left; //left child
    BinaryNode right; //right child
}
```

In C:

```c
typedef struct BinaryNode
    *PtrToNode;
typedef struct PtrToNode *Tree;

struct BinaryNode
{
    int value; //the data in the node
    BinaryNode left; //left child
    BinaryNode right; //right child
}
```
How’s That Link Work?
(how’s this differ from linked siblings?)

Can you rearrange the nodes so that a preorder traversal prints the correct sentence? Postorder?
Genetic Example

- Tree’s for showing parenthood in artificial life (AL).

In AL, you can keep track of who’s sharing artificial DNA with whom by using binary tree.
Full Binary Tree

- **Full binary tree** because all leaves have the same depth.
  - All possible spots filled in down to the same “level”.

Are you a mammal?

- no
- yes

Do you live underwater?

- yes
- no

Are you bigger than a cat?

- yes
- no

- Trout
- Robin
- Kangaroo
- Mouse
Complete Binary Tree

- **Complete binary tree** because every depth is full except the last row, which is filled from the left to some point on the right.
  - Are all full trees also complete?
  - Are all complete trees also full?

```
Are you a mammal?

Do you live underwater?

Yes

- Trout
- Robin

No

Yes

- Elephant

No

- Pink
```
Array Implementation
for complete trees ONLY!

```
[0] [1] [2] [3]
T   H   I
  S   C   L   A
 S   S   R
```

Array
Array Implementation
Now the Cool Part!

Binary Array Tree

1. Root is always in array[0]
2. If sitting on node array[i]
   - Data for a parent is always in array[(i-1)/2]
     - use integer arithmetic
   - Data for children are in array[2i+1] and array[2i+2]
     - assuming the children exist
Binary Tree ADT

- Many possibilities. These are some of the most common.
  - insert
  - delete
  - find
  - isLeaf
  - preOrderPrint, postOrderPrint
  - makeEmpty

- sometimes…
  - getLeftmost, getRightmost
  - removeLeftmost, removeRightmost
    » always traverse left (or right) until hit leaf
Example of getLeftMost

Answer: Blue. (Not Purple because that traveled right.)
When Use `getLeftMost`?

- Imagine we create a tree where the smallest numbers are always on the left. Then `getLeftMost` will get the smallest number.
  - We’re going to come back to this idea – it’s called a “binary search tree”.

```
    7
   / 
  3   11
 /   / 
1   6   23
   / 
  5
```
public int getLeftMost(BinaryNode node) {
    if(node.left == null) {
        return node.value;
    } else {
        return getLeftMost(node.left);
    }
}

Returns left-most node starting from the specified node.

sweet, oh sweet recursion!
Code for isLeaf

```java
public boolean isLeaf(BinaryNode node) {
    if(node.left == null && node.right == null ) {
        return true;
    }
    else {
        return false;
    }
}
```