Objectives

- Be able to code the nodes and references representation of a binary tree
- Be able to list and describe the member functions of class BinaryTree
- Be able to code the member functions of class BinaryTree
- Be able to build an expression tree
- Be able to write the code to evaluate an expression tree
- Be able to define the terms complete binary tree, max heap, min hep, and priority queue
- Be able to describe the sequential representation of a complete binary tree
- Be able to give the formulas for computing the location of the parent node, left child, and right child for the sequential representation
- Be able to describe the heap operations percUp and percDown

class BinaryTree

```
def __init__(self, rootValue = None)
def insertLeft(self, subtree)
def insertRight(self, subtree)
def inorderTraverse(self)
def postorderTraverse(self)
def preorderTraverse(self)
```

Building an Expression Tree

Evaluating an Expression stored as a Binary Tree

- If the root is an operand, not an operation, return its value
- Otherwise
  - Evaluate the left subtree to get leftValue
  - Evaluate the right subtree to get rightValue
  - Apply the operation at the root to get a value, and return that value

Complete Binary Trees

The sequential representation of a complete binary tree

Concept

Representation – Use only positive subscripts

Parent-child relationships

Heaps

- The term “heap” can refer to…
    - A variable-size free space list, or…
    - A particular kind of binary tree (Your text calls these ‘binary heaps’)
        - Min- vs max-heaps

Motivation – The priority queue

Some important helper methods

- percUp and percDown

Building a min heap from a stream of data

Turning an unorganized list into a max heap

Turning a max heap into a sorted list