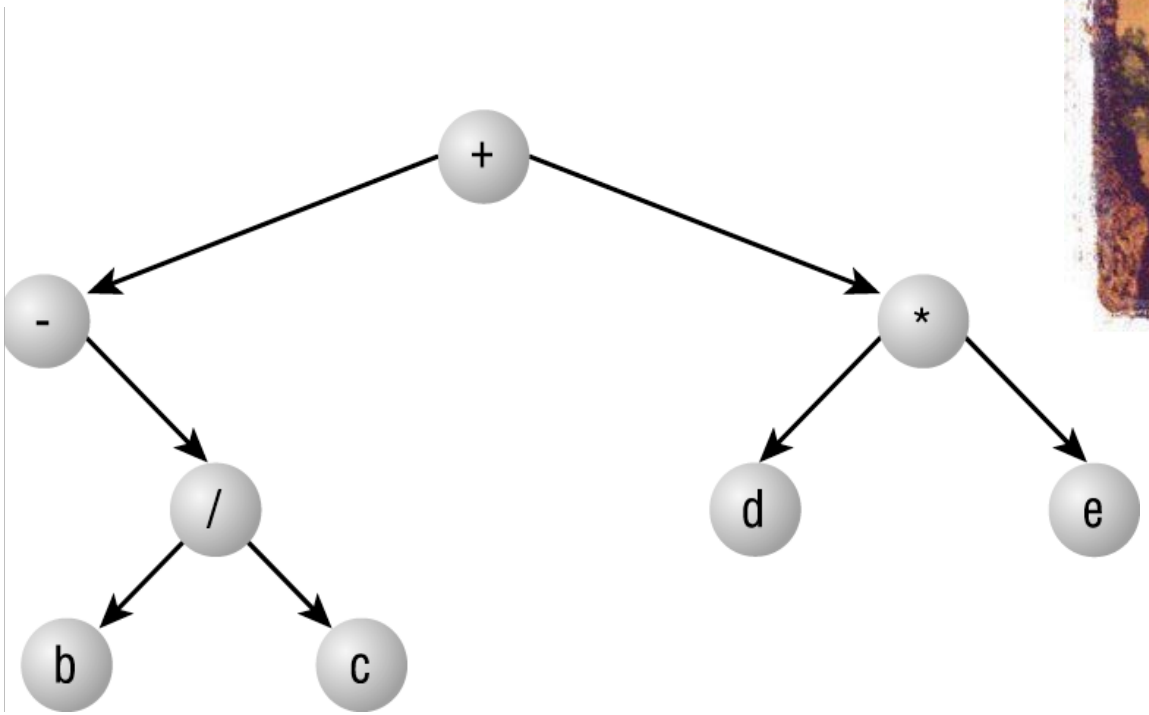
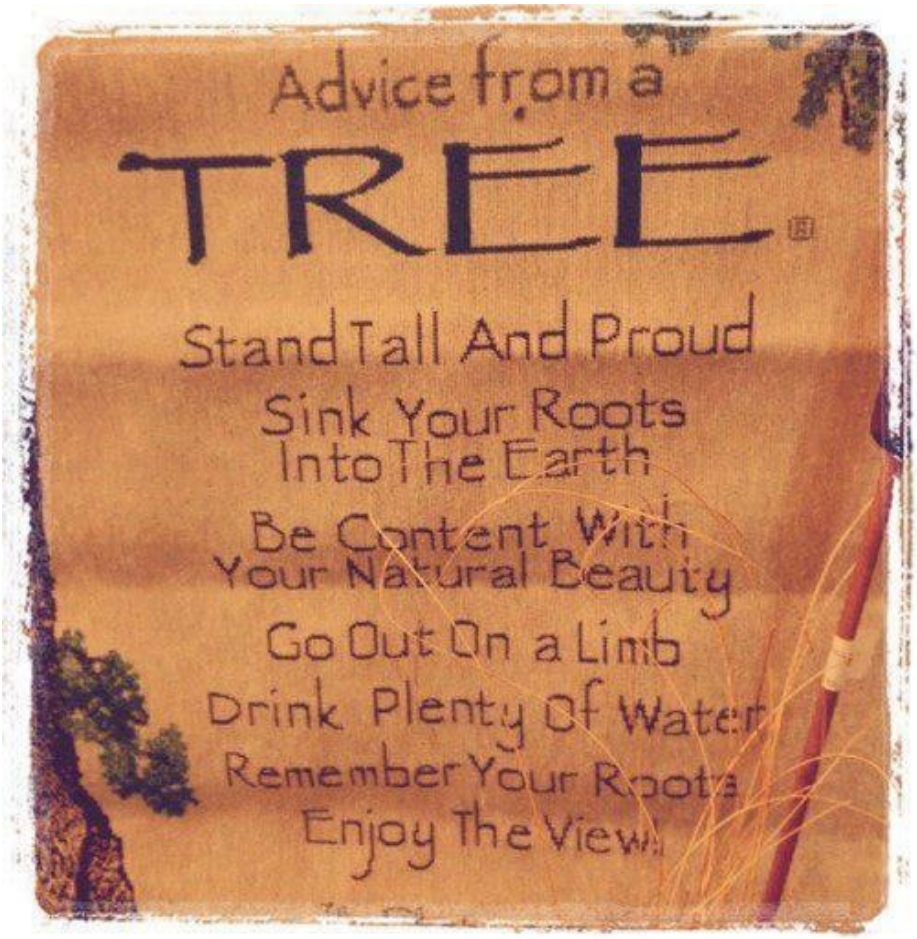
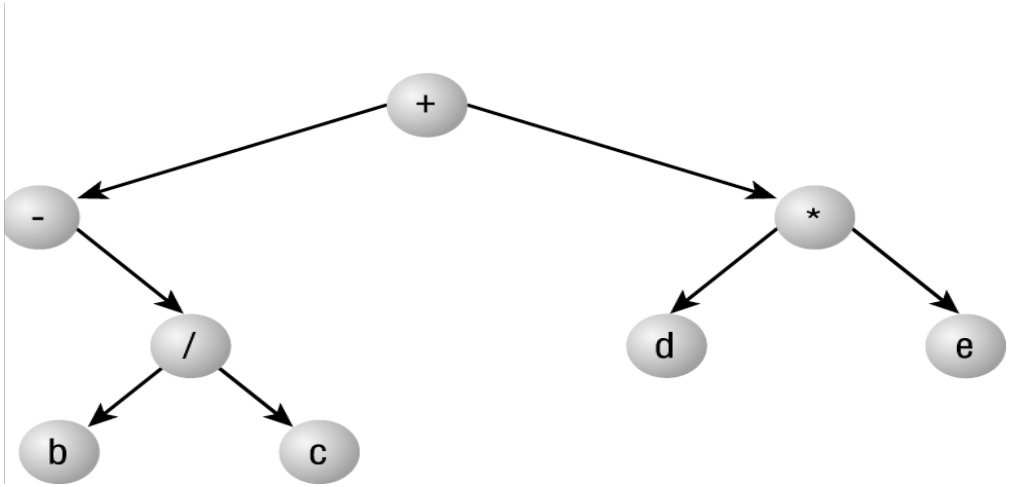
Announcements

MP4 available, due 10/16, 11:59p.





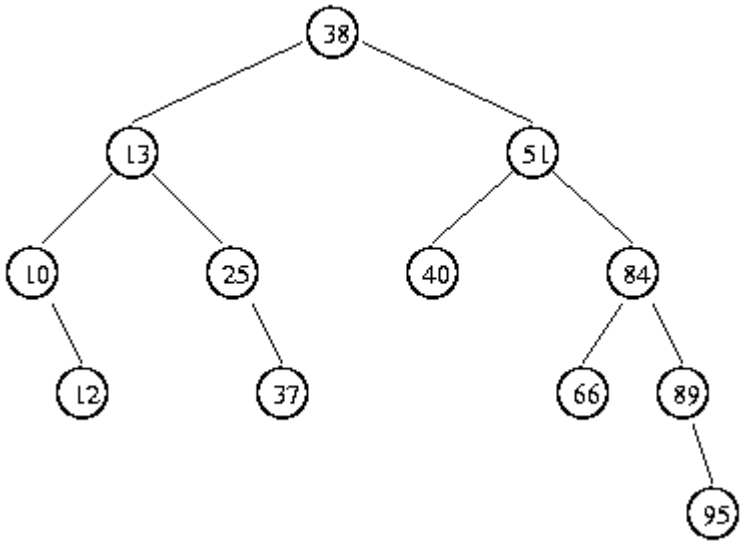
Traversals: A few discussion points…




```
template<class T>
void binaryTree<T>::preOrder(treeNode * croot){
    if (croot != null){
        yell(croot->data);
        preOrder(croot->left);
        preOrder(croot->right);
    }
}
```

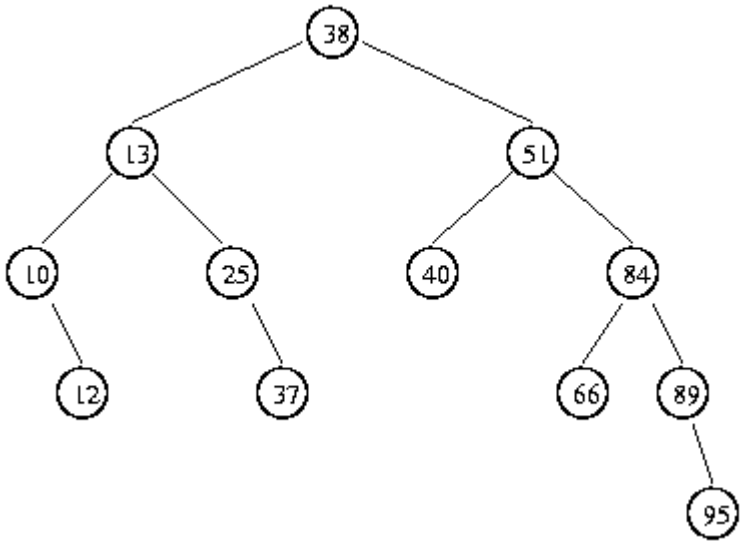
Is preOrder public or private?

Traversals: a broader view…



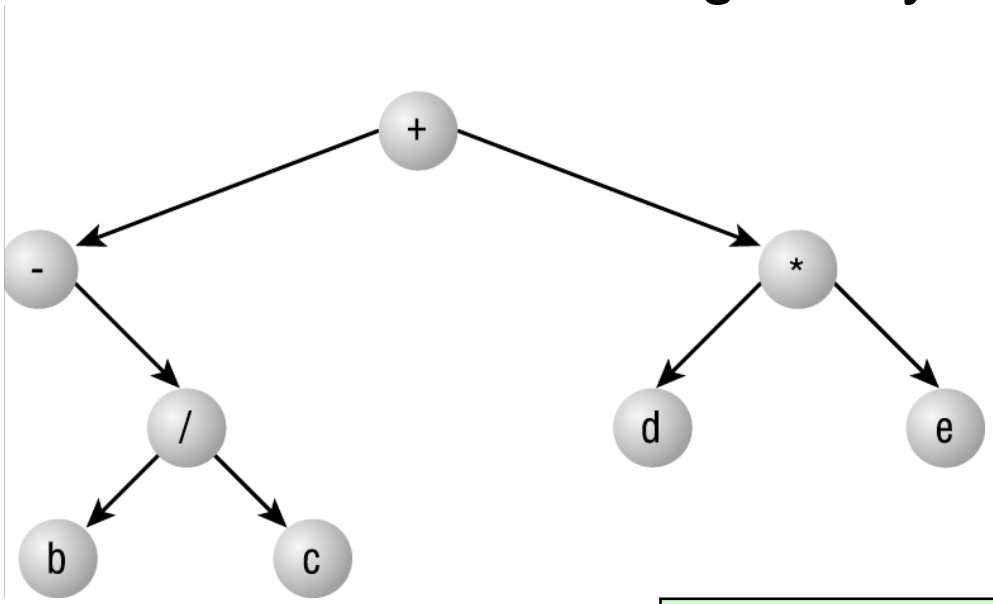

```
template<class T>
treeNode * binaryTree<T>::copy(treeNode * croot){




}
```

Traversals: another broader view…




```
template<class T>
void binaryTree<T>::clear(treeNode * croot){
  if (root != null) {
    clear(root->left)
    clear(root->right)
    delete root
    root = null
  }
}
```

Traversals: something totally different…




```
template<class T>
void binaryTree<T>::levelOrder(treeNode * croot){




}
```

Running time:

ADT Dictionary:

Suppose we have the following data…

ID #	Name
103	Jay Hathaway
92	Linda Stencel
330	Bonnie Cook
46	Rick Brown
124	Kim Petersen
…	…

…and we want to be able to retrieve a name, given a locker number.

More examples of key/value pairs:

UIN -> Advising Record

Course Number -> Schedule info

Color -> BMP

Vertex -> Set of incident edges

Flight number -> arrival information

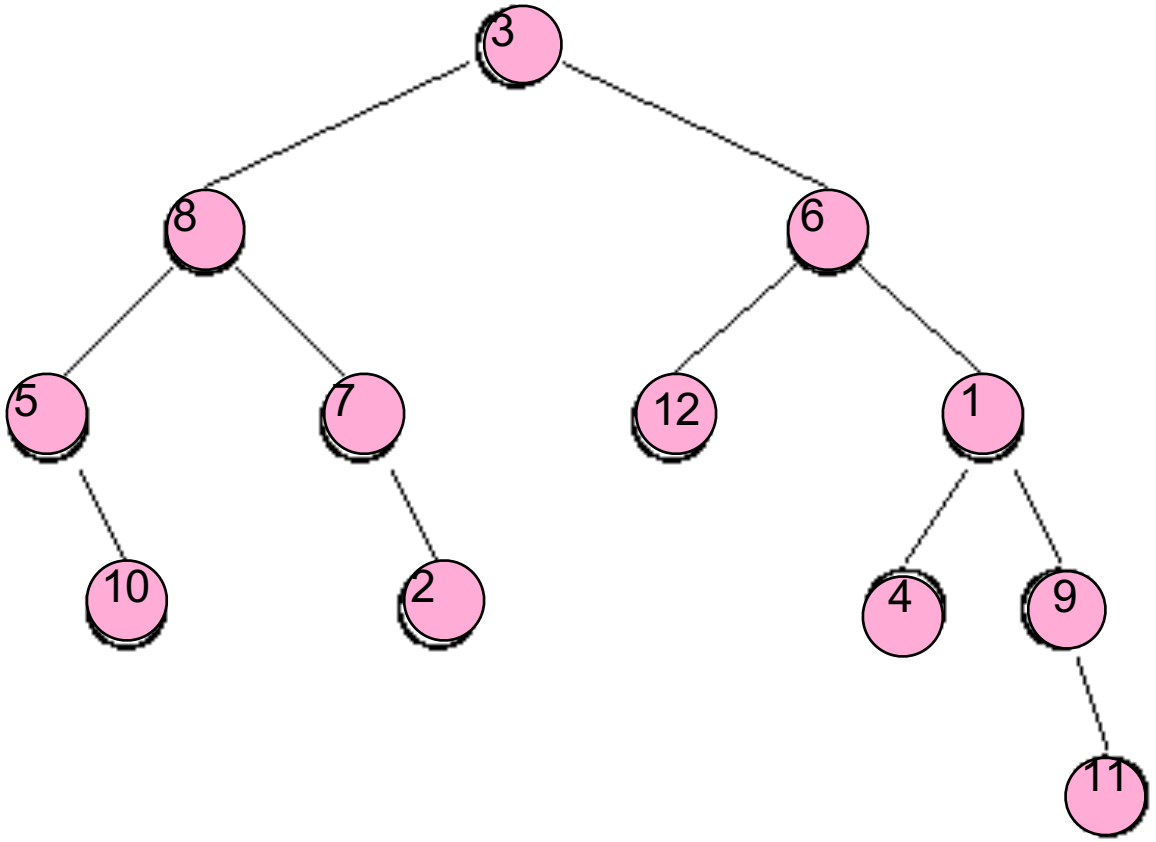
URL -> html page

A *dictionary* is a structure supporting the following:

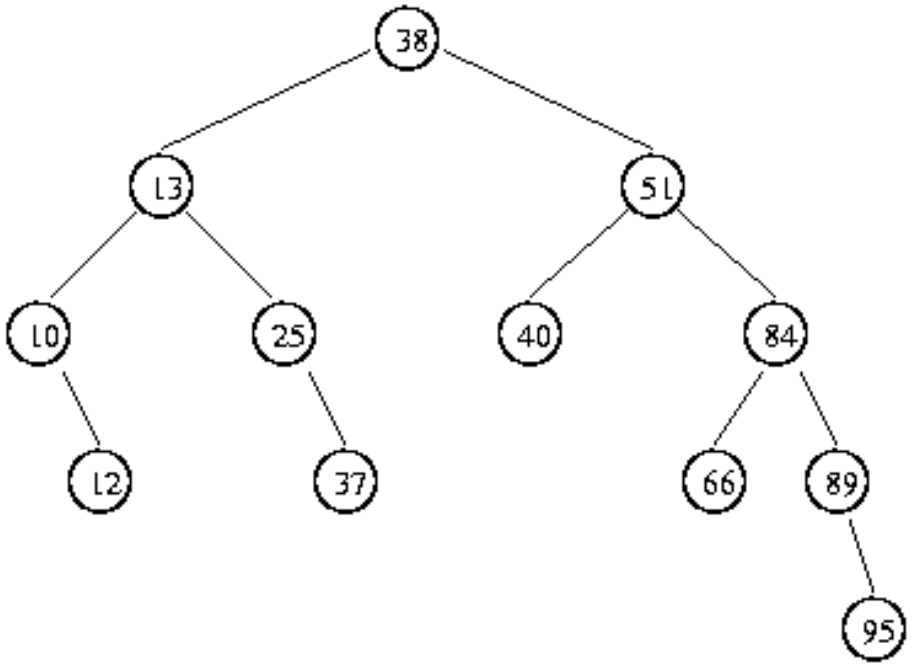
```
void insert(kType & k, dType & d)
void remove(kType & k)
dType find(kType & k)
```

Binary Trees as a search structure (Dictionary)

Find me ...



Binary _____ Tree



A Binary Search Tree (BST) is a binary tree, T, such that:

- _____, OR
- $T = \{r, T_L, T_R\}$ and

  $x \in T_L \rightarrow$ _____

  $x \in T_R \rightarrow$ _____

and