Iterator Design:

[Monday’s Lecture]: To implement an iterator, the implementing class must have two member functions:

- ::start(), returns an iterator at the first element
- ::end(), returns an iterator one past the end

```
# Iterator Design:

template <class QE>
class Queue {
public:
    class QueueIterator : public std::iterator<std::forward_iterator_tag, QE> {
        public:
            QueueIterator(unsigned index);
            QueueIterator& operator++();
            bool operator==(const QueueIterator &other);
            bool operator!=(const QueueIterator &other);
            QE& operator*();
            QE* operator->();
        private:
            int location_; 
    };
private:
    QE* arr_; unsigned capacity_, count_, entry_, exit_; 
};
```

How does the `Queue` and the `QueueIterator` interact?

Two big takeaways:

1. We will primarily talk about binary trees
2. What’s the longest “word” you can make using the vertex labels in the tree (repeats allowed)?

- Find an edge that is not on the longest path in the tree. Give that edge a reasonable name.
- One of the vertices is called the root of the tree. Which one?
- Make a “word” containing the names of the vertices that have a parent but no sibling.
- How many parents does each vertex have?
- Which vertex has the fewest children?
- Which vertex has the most ancestors?
- Which vertex has the most descendants?
- List all the vertices in b’s left subtree.
- List all the leaves in the tree.

Trees!

“The most important non-linear data structure in computer science.”
- David Knuth, The Art of Programming, Vol. 1
**Definition: Binary Tree**

A binary tree $T$ is either:

- **Tree Property: Full**

- **Tree Property: Perfect**

- **Tree Property: Complete**

---

**CS 225 – Things To Be Doing:**

1. Programming Exam A is ongoing
2. MP3 has been released; extra credit deadline is Monday!
3. lab_quacks in lab this week
4. Daily POTDs