CS 225
Data Structures

Sept. 29 – Functors
Iterators

Iterators give client code access to traverse the data!

Operators:
- `operator++`
- `operator==`
- `operator!=`
- `operator=`
- `operator*`

Types of iterators:
- Forward
- Backward
- Bidirectional

8 → 2 → 5 → Ø
#ifndef QUEUE_H
#define QUEUE_H

template <class QE>
class Queue {
    public:

    private:

};
#endif

QueueIter.h

Where does the iterator go?

What additional functions are needed for an iterator?
```cpp
#include <list>
#include <string>
#include <iostream>

struct Animal {
    std::string name, food;
    bool big;
    Animal(std::string name = "blob", std::string food = "you", bool big = true) :
        name(name), food(food), big(big) { /* none */ }
}

int main() {
    Animal g("giraffe", "leaves", true), p("penguin", "fish", false), b("bear");
    std::list<Animal> zoo;
    zoo.push_back(g);
    zoo.push_back(p);   // std::list's insertAtEnd
    zoo.push_back(b);
    for (std::list<Animal>::iterator it = zoo.begin(); it != zoo.end(); it++) {
        std::cout << (*it).name << " " << (*it).food << std::endl;
    }
    return 0;
}
```
template <class QE>
class Queue {

public:
    class QueueIterator : public std::iterator<std::bidirectional_iterator_tag, T> {
        QueueIterator(unsigned index);
        QueueIterator& operator++();
        bool operator==(const QueueIterator &other);
        bool operator!=(const QueueIterator &other);
        QE& operator*();
        QE* operator->();
    }

private:
    T* arr_; unsigned capacity_, count_, entry_, exit_;
Function Objects (aka “Functors”)

**Functors** are objects that can be called like a function.
```cpp
#ifndef DOUBLER_H
#define DOUBLER_H

class Doubler {
    public:
        int operator()(int value);
};

#endif
```

```cpp
#include "Doubler.h"

int Doubler::operator()(int value) {
    return value * 2;
}
```
template<class Value, class Modifier>
Value modify(Value value, Modifier modifier) {
  return modifier(value);
}

Doubler d;
Tripler t;
int value = 4;
modify<int, Doubler>(value, d);
modify<int, Tripler>(value, t);
This is a function called __________ whose inputs are two ______________ and a ______________.

This function appears to:
```cpp
#include <list>
#include <string>
#include <iostream>

struct Animal {
    std::string name, food;
    bool big;

    Animal(std::string name = "blob", std::string food = "you", bool big = true) :
        name(name), food(food), big(big) { /* none */ }
};

int main() {
    Animal g("giraffe", "leaves", true), p("penguin", "fish", false), b("bear");
    std::list<Animal> zoo;

    zoo.push_back(g);
    zoo.push_back(p);   // std::list's insertAtEnd
    zoo.push_back(b);

    for (std::list<Animal>::iterator it = zoo.begin(); it != zoo.end(); it++) {
        std::cout << (*it).name << " " << (*it).food << std::endl;
    }

    return 0;
}
```
Trees
Exam 4 (Programming/MP2) starts tomorrow!
More Info: https://courses.engr.illinois.edu/cs225/fa2017/exams/

MP3: Available now!
Up to +7 extra for submission by Monday, Oct. 2!

Lab: lab_quacks due Sunady, Oct 1!
Fun lab!

POTD
Every Monday-Friday – Worth +1 Extra Credit /problem (up to +40 total)