Data Structures Lab

January 20, 2011
Checkup

- Everyone should have a CS account
- You also now have a CS email account
  - username@cs.uoregon.edu
  - https://systems.cs.uoregon.edu/cube/
- Email forwarding
  - etc/procmail.settings
  - FORWARDTO=email@gmail.com
- Future listserv emails will be sent to your CS email accounts
Assignment 1 grades have been sent out
  - Check your CS email

Notes:
  - Write your names at the top of your files
  - Make sure you match the sample input and output
  - Your code should work for arbitrarily sized inputs
  - Comment your code
General Coding Guidelines

- This is a data structures lab
  - You are here to code data structures
  - Don't use built in data structures!

- I don't want to see the following in your code
  - #include<vector>
  - #include<queue>
  - #include<anything other than iostream or string>
General Coding Guidelines

- Your code should work for arbitrarily large inputs
  - Don't reject inputs that are "too large" for your code
  - I don't care if you reject them gracefully

- Fixed size arrays are not scalable
  - Don't use them unless you know they will be sufficient
  - (or you have a plan for when they fill up)
General Coding Guidelines

- Main methods should only contain run-time code
- Classes should be declared in their own space
- Static methods should be declared in their own space
- Don't put all your code in your main method!
General Coding Guidelines

- Comment your code
  - Name and other useful info at the top
  - Description before each method
  - Inline comments next to complicated lines

```c
/*
 * Greg Bickerman
 * Assignment 1
 * Stack header file
*/
```

- You may use your code in future assignments
  - You'll want to know what you were doing

- If you code goes wrong, I will want to know what you were doing
General Coding Guidelines

- Use header files
- Use them as outlines when coding
- Your compiler can use them as checklists
  - If you're missing a method, your compiler will let you know
Including Header files

Stack.h

... code ...

-------------------------------------------------------

Stack.cpp

#include "Stack.h"

... code ...

Including Header files

Stack.h

```c
#ifndef STACK_H
#define STACK_H
...
code...
...#endif
```

Stack.cpp

```c
#include "Stack.h"
...
code...
...```
Homework 1 - The Josephus Problem

● Due Friday January 21 (tomorrow)

● The Problem
  o n soldiers in a circle
  o Kill off every other soldier
  o Who is left at the end?
Homework 1 - The Josephus Problem
Homework 1- The Josephus Problem
Homework 1 - The Josephus Problem
Homework 1- The Josephus Problem
Homework 1- The Josephus Problem
Homework 1 - The Josephus Problem
Homework 1 - The Josephus Problem
Homework 1 - The Josephus Problem
Homework 1 - The Josephus Problem

- What data structure could help us solve this problem?
- Might require some circular reasoning...
Circular Linked List

- Just like an ordinary linked list
  - But the last node connects to the first node

- What data members will we need?
  - (What is the head of a circular linked list?)

- What methods will we need?
  - (How should elements be inserted and deleted?)
Circular Linked List

- **Node**
  - string name
  - Node* next

- **Circle:**
  - Member variables:
    - Node* current
    - int count
  - Methods:
    - void insert(string name)
    - void kill()
    - string getCurrent()
Circular Linked List

- **Node**
  - `string name`
  - `Node* next`

- **Circle**:
  - **Member variables:**
    - `Node* current`
    - `int count`
  - **Methods:**
    - `void insert(string name)`
    - `void kill()`
    - `string getCurrent()`

- Now turn it into a header file
struct Node {
    string name;
    Node* next;
};

class Circle {
    Node* current;
    int count;

    Circle();
    void insert(string name);
    void kill();
    string getCurrent();
};
# Circle.h

```cpp
struct Node {
    string name;
    Node* next;
};

class Circle {
    Node* current; //The current "head" of the circle
    int count; //The number of nodes in the circle

public:
    Circle(); //Constructor
    void insert(string name); //Insert a node into the circle
    void kill(); //...
    string getCurrent(); //Return the current "head" of the circle
};
```
Assignment 2

- Due Friday, February 4th
  - Two weeks

- Binary Trees
  - Have your cake and sort it too

Heigh Ho...
Assignment 2

- The seven dwarves have just returned home with a cart of diamonds
  - (all of conveniently integer weights)

- Unfortunately, the wicked witch has just imposed a diamond tax
  - Each tax specifies a weight \( w \)
  - It must be paid with a diamond weighing at least \( w \)

- If the dwarves can't pay the witch's tax, she'll repossess their house
Assignment 2

<table>
<thead>
<tr>
<th>Diamonds</th>
<th>Taxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Assignment 2

Diamonds

10
5
12
3

Taxes

5
8
11
Assignment 2

Diamonds

- 10
- 5
- 12
- 3

Taxes

- 6
- 8
- 11
Assignment 2

Diamonds

10
5
12
3

Taxes

6
8
11

Uh Oh
Homework 2

- Solutions are not trivial

- How can we sort through sets of arbitrary numbers
  - Need some data structure that imposes order

- What do we need to do?
  - Insert
  - Find
  - Delete
Binary Trees to the rescue

More info next week...