Data Structures Lab

Under the Hood
Checkup

● You should all be on the CIS 323 listserve
  ○ Check your CIS email
  ○ https://systems.cs.uoregon.edu/cube/
  ○ If you didn't get a listserve email, let me know

● Warmup grades have been sent out
  ○ Check your CIS email
  ○ If you didn't receive a grade, let me know
Warmup Assignment Notes

- General issues
  - Write your names at the top of your files
  - Run the I/O tests I provide for you
  - Make sure you match the sample input and output
  - Comment your code

- No pair programmers
  - Upcoming assignments will be harder
  - Your classmates are friendly
C++ Spotlight - Memory Management

- I've posted some stack sample code
  - Hopefully people have looked at it
  - Hopefully it doesn't look like greek

- Most C++ code is looks very similar to java
  - Minor syntax aside...

- There are a number of subtle differences
  - What's a \texttt{Node*}?
  - Why (and when) do I need to \texttt{delete} things?
C++ Spotlight - Memory Management

- Two areas of memory
  - Stack memory
  - Heap memory
int main(){
    int a = 5;
    int b = 6;
    int c = plus(a,b);
    int d = times(a,b);
}

int plus(x,y){
    return x+y;
}

int times(x,y){
    return x*y;
}
int main(){
    int a = 5;
    int b = 6;
    int c = plus(a, b);
    int d = times(a, b);
}

int plus(x, y){
    return x + y;
}

int times(x, y){
    return x * y;
}
int main(){
    int a = 5;
    int b = 6;
    int c = plus(a,b);
    int d = times(a,b);
}

int plus(x,y){
    return x+y;
}

int times(x,y){
    return x*y;
}
int main()
{
    int a = 5;
    int b = 6;
    int c = plus(a, b);
    int d = times(a, b);
}

int plus(x, y)
{
    return x + y;
}

int times(x, y)
{
    return x * y;
}
int main()
{
    int a = 5;
    int b = 6;
    int c = plus(a, b);
    int d = times(a, b);
}

int plus(x, y)
{
    return x + y;
}

int times(x, y)
{
    return x * y;
}
int main(){
    int a = 5;
    int b = 6;
    int c = plus(a,b);
    int d = times(a,b);
}

int plus(x,y){
    return x+y;
}

int times(x,y){
    return x*y;
}
int main(){
    int a = 5;
    int b = 6;
    int c = plus(a, b);
    int d = times(a, b);
}

int plus(x, y) {
    return x + y;
}

int times(x, y) {
    return x * y;
}
Memory Management - The Stack

● What's wrong with this code?

```c
void push(int i){
    Node n = Node(i);
    n->next = this->head;
    this->head = n->next;
}
```
Memory Management - The Stack

● What's wrong with this code?

```cpp
void push(int i){
    Node n = Node(i);
    n->next = this->head;
    this->head = n->next;
}
```

● The new node is declared on the stack
  ○ When it goes out of scope, it will be lost!
Memory Management - The Heap

- The `new` keyword declares an object on the heap
  - And returns a pointer to it

```c
int main()
{
    int a = int(5);
    int* b = new int(6);
}
```
The `new` keyword declares an object on the heap
  ○ And returns a pointer to it

```cpp
void push(int i){
    Node* np = new Node(i);
    np->next = this->head;
    this->head = np->next;
}
```

When the insert function returns...
  ○ `np` will go out of scope
  ○ But the node in the heap will remain
Getting to the Point

- C++ differentiates between objects and pointers to objects

- Node n = Node(42);
  - Initialized in stack memory
  - n is a stack object
  - n will be deleted when it goes out of scope

- Node* np = new Node(42);
  - A Node is initialized in heap memory
  - np is a pointer to that Node object
  - The node will remain indefinitely
Getting to the Point

● What do you do with a pointer?

● Dereference it
  ○ Node* np = new Node(5);
  ○ Node myNode = *np;

● Follow it
  ○ np->data;
  ○ syntactic sugar for (*np).data

● Increment it?
  ○ ++np;
  ○ Generally a bad idea...
Taking out the Trash

- Stack objects disappear when they go out of scope
  - That's how the stack works

- What about heap objects?

- Some languages perform garbage collection
  - Search through the heap for "garbage"
  - Delete objects which are no longer in use

- C++ doesn't do anything for you unless you ask
  - Maybe you wanted to leave that node object there...
Taking out the Trash

● If you create an object, you're responsible for deleting it
  ○ Node* np = new Node();
  ○ delete np;

● delete is the opposite of new
  ○ new creates an object and gives you a pointer to it
  ○ delete takes a pointer and deletes its object

● For every call to new, there should be a call to delete
  ○ Otherwise, you get memory leaks
Homework 1 - The Josephus Problem

- Due Friday January 21 (tomorrow)

- The Problem
  - n soldiers in a circle
  - Kill off every other soldier
  - 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

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

● Node
  ○ string name
  ○ Node* next

● CLinkedList:
  Member variables:
  ○ Node* head
  ○ int count
  Methods:
  ○ void insert(string name)
  ○ string remove()
  ○ void shift()
  ○ string getCurr()
class CLinkedList {
    Node* head;  //The current "head" of the list
    int count;   //The number of nodes in the list

    public:
    CLinkedList();
    void insert(string);
    string remove();
    void shift();
    string getCurr();
};
Chalkboard Time

Let's implement insert
Odds and Ends

- Insert and remove should run in constant time
  - Don't cycle all the way around the list

- Make sure to test your data structure
  - Might want a size method
  - Might want a print method

- If the Linked List is broken
  - No credit on the Josephus component

- Any lingering questions?
Assignment 2

- Due Friday, February 10th
  - Two weeks

- Binary Trees
  - Have your cake and sort it too

Heigh Ho...
Assignment 2

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

- Unfortunately, the wicked witch has 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

Diamonds

- 10
- 5
- 12
- 3

Taxes

- 5
- 8
- 11
Assignment 2

Diamonds

10
5
12
3

Taxes

5
8
11
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

<table>
<thead>
<tr>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Taxes

<table>
<thead>
<tr>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
</tr>
<tr>
<td>11</td>
</tr>
</tbody>
</table>
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
  ○ Delete
  ○ Find
  ○ Next
Binary Trees to the rescue

Stay tuned...