Interactive Lecture Questions

- Ask Questions: Ask in-lecture questions using this Google Form! Questions are reviewed and answered live during lecture.
- Detailed Answers After Lecture: If you didn't get to answer your question in lecture, we provide detailed answers to common questions here.
- You must be logged in with an @illinois.edu Google account. If you get access denied, open the link in a private tab and be asked to log in.

Lecture Videos

- Recorded on echo360.org, log in with your @illinois.edu e-mail address

Schedule

<table>
<thead>
<tr>
<th>Monday</th>
<th>Wednesday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 15</td>
<td>January 17</td>
<td>January 19</td>
</tr>
<tr>
<td>MLK Day</td>
<td>Intro</td>
<td>Classes</td>
</tr>
<tr>
<td></td>
<td>slides</td>
<td>handout</td>
</tr>
<tr>
<td>January 22</td>
<td>January 24</td>
<td>January 26</td>
</tr>
<tr>
<td>Memory</td>
<td>Heap + Parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>slides</td>
<td>handout</td>
</tr>
</tbody>
</table>
Interactive Lecture Questions

- Ask Questions: Ask in-lecture questions using this Google Form!
- Detailed Answers After Lecture: If we didn't get to answer your question in lecture, we provide detailed answers to common questions here.
- You must be logged in with an active CS 225 account to be asked to log in.

Lecture Videos

- Recorded on echo360.org, login with active CS 225 account

Schedule

Monday

January 15
MLK Day

January 22
Memory

CS 225 - Lecture Questions

Your email address (waf@illinois.edu) will be recorded when you submit this form. Not you? Switch account

* Required

Question for Lecture: *

Your answer

Submit

Never submit passwords through Google Forms.
Interactive Lecture Questions

- Ask Questions: Ask in lecture questions using this Google Form! Questions are reviewed and answered live during lecture.

Detailed Answers After Lecture: If we didn't get to answer your question in lecture, we provide detailed answers to common questions here. You must be logged in with an @illinois.edu Google account. If you get access denied, open the link in a private tab and you should be able to log in.

Lecture Videos

- Recorded on echowith

Schedule

Monday

January 15
MLK Day

January 22
Memory slides | handout | pointers

1/31/2018 – Lecture: Inheritance

1. When do we use the heap memory?
   
   https://www.gribblelab.org/CMBootCamp/7_Memory_Stack_vs_Heap.html

   If you need to allocate a large block of memory (e.g. a large array, or a big struct), and you need to keep that variable around for a long time (and in different functions), then you should allocate it on the heap. If you are dealing with relatively small variables that only need to persist as long as the function using them is alive, then you should use the stack, it’s easier and faster. If you need variables like arrays and structs that can change size dynamically (e.g. arrays that can grow or shrink as needed) then you will likely need to allocate them on the heap.

2. Operators overloading and how to use them:
   
   Useful links:
   
   https://www.tutorialspoint.com/cppplus/cpp_overloading.htm
   https://www.geeksforgeeks.org/operator-overloading-c/

3. How does overloading [ ] or [] work? What do you do with it?
   
   https://www.geeksforgeeks.org/overloading-subscript-or-array-index-operator-in-c/
Lecture Resources

January 19
Classes
slides | handout | code | TA Notes

- C++ program starts with calling main and in fact, main is the only function called automatically. The rest of the functions are called from the main.
- In the main and C++ we include two files differently. When we include a file, we are including the contents of that file. When we #include a file, we are including a copy of that file indicating that the code belongs to the file. The #include operator includes a file into the program.
- A header file is a special kind of file. The header files are stored in the standard library, which we included in the program. The include operators are a special kind of preprocessor directive that tells the compiler to include the file.
- A main function has a special kind of keyword main, which tells the system that this is the entry point of the program. The main function is the first function to be called in a C++ program.
template <class QE>
class Queue {
    public:
        class QueueIterator : public std::iterator<std::bidirectional_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_; 
};
Big Ideas

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

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

A tree is:

•

•
A Rooted Tree

“Mario Family Line”
<http://limitbreak.gameriot.com/blogs/Caveat-Emptor/Mario-Family-Line>
More Specific Trees

We’ll focus on **binary trees**:

- A binary tree is **rooted** – every node can be reached via a path from the root.
More Specific Trees

We’ll focus on **binary trees**:  
- A binary tree is **acyclic** – there are no cycles within the graph
More Specific Trees

We’ll focus on **binary trees**: 

- A binary tree contains **two or fewer children** – where one is the “left child” and one is the “right child”: 
Tree Terminology

• What’s the longest “word” you can make using the vertex labels in the tree (repeats allowed)?
Tree Terminology

- 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 an “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 is b’s left subtree.
- List all the leaves in the tree.
Tree Terminology

- 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 an “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.
Tree Terminology

• 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 an “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.
Tree Terminology

• 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 an “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.
Tree Terminology

• 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 an “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.
Binary Tree – Defined

A binary tree $T$ is either:

- OR
- 
Tree Property: height

*height(T)*: length of the longest path from the root to a leaf

Given a binary tree T:

\[ \text{height}(T) = \]

```
  C
 /   \
S     X
   /   \
  A   2
     /   \
    2   5
```
Tree Property: full

A tree $F$ is **full** if and only if:

1. 
2. 
Tree Property: perfect

A perfect tree $P$ is:

1.

2.
Tree Property: complete

**Conceptually**: A perfect tree for every level except the last, where the last level is “pushed to the left”.

**Slightly more formal**: For any level $k$ in $[0, h-1]$, $k$ has $2^k$ nodes. For level $h$, all nodes are “pushed to the left”.
Tree Property: complete

A complete tree $C$ of height $h$, $C_h$:

1. $C_{-1} = \{\}$

2. $C_h$ (where $h > 0$) = $\{r, T_L, T_R\}$ and either:

   $T_L$ is ___________ and $T_R$ is ___________

   OR

   $T_L$ is ___________ and $T_R$ is ___________
Tree Property: complete

Is every full tree complete?

If every complete tree full?