Checkup

- Does everyone have a CS account?
- Did everyone get the listserv email?
- Can everyone write compile and run C++ files?
- Has everyone looked at the homework?
  - (Due tomorrow at midnight)
Survey Results

- **Language**
  - 50% Haven't seen C++ before
  - 20% Have seen C++ briefly

- **Experience**
  - Less then 1 year
  - Coding since high school

- **Expectations**
  - Learn C++
  - Apply concepts from CIS 313
  - Practice coding
Helpy Hour

- Deschutes 100
- Tuesday, Wednesday 3 - 5 PM
  - (Stay after on Tuesday for HELP)
- Run by ACM student volunteers
Notes on Warmup Assignment

Make sure your code conforms exactly to the assignment

INPUT:
7
Bashful
Doc
Dopey
Grumpy
Happy
Sleepy
Sneezy

OUTPUT:
Hello, Bashful!
Hello, Doc!
Hello, Dopey!
Hello, Grumpy!
Hello, Happy!
Hello, Sleepy!
Hello, Sneezy!
Notes on Warmup Assignment

Make sure your code conforms exactly to the assignment

<table>
<thead>
<tr>
<th>INPUT:</th>
<th>OUTPUT:</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Input a Number:</td>
</tr>
<tr>
<td>Bashful</td>
<td>Hello Bashful</td>
</tr>
<tr>
<td>Doc</td>
<td>Hello Doc</td>
</tr>
<tr>
<td>Dopey</td>
<td>Hello Dopey</td>
</tr>
<tr>
<td>Grumpy</td>
<td>Hello Grumpys</td>
</tr>
<tr>
<td>Happy</td>
<td>Hello Happy</td>
</tr>
<tr>
<td>Sleepy</td>
<td>Hello Sleepy</td>
</tr>
<tr>
<td>Sneezy</td>
<td>Hello Sneezy</td>
</tr>
</tbody>
</table>
Notes on Warmup Assignment

- Make sure your code compiles
- Make sure your output matches the assignment exactly
- Make sure your code is commented
- Always compile and test your code right before submitting
  o (even if you only added one minor comment)
Why use the terminal?

- Different projects use different development platforms
  - But everyone uses the same terminal
- Fast, lightweight coding
- That's how I'll be grading your code
Moving around the terminal

- "cd dir" - change directory to *dir*
  - "cd .." - move up one directory
  - "cd ~" - move to home directory

- "ls" - list files in current directory
  - displays both files and folders

- "pwd" - display current directory
  - in case you forget where you are
Command line compilation

- "g++ filename"
  - compiles to a.out

- "g++ filename -o binaryname"
  - compiles to binaryname

- "./binaryname"
  - runs your binary

Example:
  - g++ helloworld.cpp -o hello
  - ./hello
Command line I/O

- Usually, "cout << text" prints out text to standard output
  - What if we want to output to a file?

- ./hello > out.txt
  - Writes the output of hello to out.txt

- Similarly, "cin >> input" stores text from standard input in input
  - What if we want to automatically input text?

- ./hello < in.txt
  - Uses in.txt as standard input
Why is this useful?

● Testing!
  o I like to give you sample input/output
  o Let the computer check them for you

● You can use "<" and ">" at the same time
  o "./assignment < input.txt > output.txt"

● Checking your output is just as easy
  o "diff file1 file2" prints out differences between two files
  o "diff output.txt expected.txt"
Data Structures!

- What exactly is a data structure?
Data Structures!

- What exactly is a data structure?
  - A structure for storing data

- Allows program to
  - Access data
  - Insert data
  - Delete data
Arrays: The primitive data structure

- Stores elements sequentially in memory

- Pros:
  - It's built in
  - Practically no overhead

- Cons:
  - Fixed size
  - No structure
Can we do better?

- Elements of arrays are fixed in memory.
- If we could store elements in arbitrary memory locations, we could store arbitrarily many elements.
- But we'd need some way of remembering where they were located...
Linked Lists

- Collection of nodes
- Each node is responsible for keeping track of its neighbors
- The list as a whole only needs to keep track of one node
Linked Lists
Linked Lists

- Store data in nodes
- How do you access data in a linked list?
- How do you insert into a linked list?
- How do you delete from a linked list?
Linked Lists

- Queue
  - FIFO - First In First Out
  - Enqueue through the back
  - Dequeue from the front

- Stack
  - FILO - First In Last Out
  - Push onto the front
  - Pop from the front
Let's make a stack

- What will it look like?
Stack - Design

Node

Data = 27    Next
Stack - Design

Stack
Head Size = 3

27 ➔ 45 ➔ 19
Stack - Design

- **Node**
  - Member Variables:
    - Int data
    - Node* next

- **Stack**
  - Member Variables:
    - Node* head
    - Int size

- What methods will we need?
Stack - Design

- **Node**
  - **Member Variables:**
    - Int data
    - Node* next

- **Stack**
  - **Member Variables:**
    - Node* head
    - Int size
  - **Methods:**
    - void push(int n)
    - int pop()
    - int top()
    - int size()
Stack.h

struct Node {
    Int data;
    Node* next;
};

class Stack{
    Node* head;
    Int size;

public:
    void push(int n);
    int pop();
    int top();
    int size();
};
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

- 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
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

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

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

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

- Now turn it into a header file