# Linked Structures, Project 1: Linked List

Bryce Boe

2013/10/16

CS24, Fall 2013

## Outline

- Lab 3 Review
- Project 1 Notes
- Linked Structures

## LAB 3 REVIEW

## Student Comments

- “I haven't learned anything new but instead have spent hours writing a variety of loops and random code searching for some bug…”
- “What I've been doing was unceasingly putting in random functions and random parameters for the whole day trying to figure out some bugs.”
- “And the worse thing is that I don't even know how it came out when I found it…”

## Lab 3 was supposed to be challenging

- Many of you practiced fuzz testing
  - Great way to perform black-box testing to attempt to find bugs
- An exercise in trying something and testing if it worked
  - Code, compile, test (repeat over and over)
- Now we'll make sense of the bugs

# Project 1 Notes

- Do not use **static** or **global** variables (you will lose significant points if you rely on them)
- You must validate all memory allocations
  - if an allocation fails make sure you don't leak other memory
- Get the array list working up to size N before dealing with reallocation

## LINKED STRUCTURES

## What’s wrong with using arrays to store data?

- Arrays require continuous chunks of memory
- Unless the array is full, there is *wasted* space
- Expanding the array is typically done by doubling the size
  - Worst case time: Have to copy all the existing items
  - Hint: realloc does this for you (recall how realloc is implemented)

## How long does it take?

- Appending an item to a non-full array?
- Appending an item to a full-array?
- Removing an item from the end of the array?
- Removing an item from the beginning of the array?
- Accessing an element in the middle of the

## Single-link **Node** structure

```
struct Node {
	int _data;
	struct Node *_next;
}
```

## Node allocation walkthrough

- Add an initial node
- Add another node at the beginning
- Add another node at the end
- Remove a node at the beginning
- Remove a node at the end

## How can we access the data of specific elements?

- The data of the second element?
  - head->_next->_data
- The data of the third element?
  - head->_next->_next->_data;
- (Generally) The data of the nth element?

```
struct Node *tmp = head;
for (int i = 0; i < n; ++i)
        tmp = tmp->_next;
tmp->_data;
```