CSE 333 – SECTION 2

Manual Memory Management: Mastering `malloc()`

Office Hours

- Hal
  - Mondays, 4pm-5pm (CSE 548)
  - Or by appointment
  - Or just drop in if the door's open
- Renshu
  - Tuesdays, 3:30pm-4:30pm (CSE 006)
- Johnny
  - Wednesdays, 3:30pm-4:30pm (CSE 006)
- Sunjay
  - Thursdays (that's today!), 3:30pm-4:30pm (CSE 006)
- Cortney
  - Fridays, 3:30pm-4:30pm (CSE 006)
- Discussion board!
  - All day, every day

Questions, Comments, Concerns

- Do you have any?
- Exercises going ok?
- Lectures make sense?
- Looked at the homework?

Using the Heap

- Why is this necessary?
- Lifetime on the stack
- Lifetime on the heap

Memory Management

- C gives you the power to manage your own memory
- C does very little for you
- Benefits? Disadvantages?
- When would you want this vs. automatic memory management?

Memory Management Done Right

- Need to let the system know when we are done with a chunk of memory
- In general, every `malloc()` must (eventually) be matched by a `free()`

- Example:
- **`[lec04_code/arraycopy.c]`**

Memory Management Details

- When are we done with a piece of data?
- Depends on where we got it from, how we are using it, etc.
- Some functions expect allocated space, others allocate for you
  - `sprintf()` vs `asprintf()`
- Some APIs expect you to free structures, others free for you
  - Compare / contrast?

Memory Management Gone Horribly Wrong

- Many (many!) ways to mess up
- Dangling pointers
- Double frees
- Incorrect frees
- Never frees
- That's just a few
- Small example: `[badlylinkedlist.c]`

Valgrind Is Your Friend

- Use of uninitialized memory
- Use of memory you shouldn't be using
- Memory leaks
  - Definitely Lost
  - Indirectly Lost
  - Possibly Lost
  - Still Reachable*

- Simply run: `valgrind <program>`
- Small example: `[imsobuggy.c]`

*This is generally ok