CS50 Section. Week 7. 10/20/15.

Tuesdays 7:00-8:30pm, Science Center 309A
https://github.com/hathix/cs50-section

Neel Mehta
neelmehta@college.harvard.edu
(215) 990-6434

Get these handouts at https://github.com/hathix/cs50-section/tree/master/handouts.

pset5

USE HASHTABLES, NOT TRIES! I tried using a trie last year (pun intended) and it was really painful. Hashtables are way easier and tend to run faster.

Choosing a good hash function

You'll need a hash function for `load` and `check` -- something to assign words to buckets in your hashtable. Some specifications:

- It must have a header like this: `unsigned int hash(char* word)`
- It must return an index that's less than the number of buckets, `BUCKETS` (mod `%` by `BUCKETS` to ensure this)

Don't try writing your own hash function... just adapt one off the internet. Some places you can look:

- My compilation of hash functions from around the internet: https://github.com/hathix/cs50-section/blob/master/code/7/sample-hash-functions
- Random Stack Overflow questions like http://stackoverflow.com/q/7666509 (search for "string hash functions for c")

Random advice gathered from past TFs (might be a little advanced)

- Use `#define BUCKETS` for the number of buckets in your hashtable
- Check for `NULL` when you're using functions that might fail (e.g. `fopen`, `malloc`)
- Always `fclose` your files
- Run valgrind with `show-reachable=yes`
- Use `mmap` and `calloc` when allocating memory (better than plain ol' `malloc`)
- Avoid using `strlen` when possible because it's slow; if you're using a loop try to just compare each character

to `\0`
- Eliminate function calls when possible because they take up lots of time

More coming soon!