Week 1

* Floating point mantissa runs out of memory when you add onto long precise floats
* Unix is a stripped down Multix that pretends: **everything is a file** & **Each command should do one thing very well**
* **OS X** is a version of Unix!!!
* Unix cmd:
  + cd changes dir
  + cd .. back
  + cd . refreshes current dir
  + cd ~ home dir
  + cd / root dir
  + pwd prints current dir
  + mkdir makes dir
  + rmdir removes / deletes dir
  + cat concatenates two files
  + emacs, nano, and vi let you create / edit files
  + vi [filename.filetype]
  + EXIT\_SUCCESS = 0 = everything went well
  + EXIT\_FAILURE = 1 = shit.
  + **./**[filename] **RUNS YOUR FILE**
* C:
  + It doesn’t know how long arrays are
  + It doesn’t have a String class, but **it has pointers to chars!**
  + Add \* to the end of a type declaration to define that var as a pointer to that type
  + C string functions:
    - Use printf(“[one of the below function]”, [var of the specified type])
    - %d : int in decimal
    - %x : int in hex
    - %X : int in HEX
    - %g : int in floating point
    - %f : int in floating point
    - %s : int in String
    - %c : int in char
  + C switch statements:
    - switch (**ITEM**)
    - {
      * case **CASE** :
        + \_\_\_\_
        + break;
      * case **CASE** :
        + \_\_\_\_
        + break;
      * case **CASE** :
        + \_\_\_\_
        + break;
      * default :
        + \_\_\_\_
    - }
  + C Input:
    - Use scanf(“%d”, &**[VARIABLE TO STORE VALUE TO]**)
    - Use sscanf(“%d”, &**[VARIABLE TO STORE VALUE TO]**) for error checking
    - fgets
    - atoi
    - atof
  + C pointers:
    - If you have an int
      * &[var] gets you the pointer value
      * [var] gets you the int value
    - If you have an int pointer
      * &[var] gets you the pointer value
      * [var] gets you the pointer value
      * \*[var] gets you the int value