CS 16 6/6 Review

* Topics covered in “Kevin’s Notes”
	+ (4/30) Arrays, Binary Search, Call by Reference, Pointers
	+ (5/2) More on Pointers, Arrays
	+ (5/7) Structs, Pointer arithmetic, LLDB
	+ (5/9) Const, more on pointer arithmetic, new and delete, memory leak, dangling pointer
	+ (5/14) Infinite inclusion, Linked Lists, Recursion, Helper Functions
	+ (5/21) Recursion, Helper Functions
	+ (5/23) Strings, Helper Functions and Recursion
	+ (5/28) Strings, Fibonacci, Binary Trees
* All Boolean values other than 0 are true.
* Do-While loops execute the body before checking the condition.
	+ Do-While executes at least once
* While loops check the condition before executing the body.
	+ While loops may never execute
* If we do x= 5; y = x++; then x = 6, but y = 5.
	+ If we do x= 5; y = ++x; then x = 6, and y = 6.
	+ Saying y = x++ is equivalent to saying y = x; x+=1;
	+ Saying y = ++x is equivalent to saying y = x += 1;
* Data Representation
	+ Ints and Pointers are 4 bytes
	+ Doubles are 8 bytes
	+ Chars are 1 byte
* Conversion from decimal to binary
	+ Conversion from binary to hexadecimal
	+ Conversion from decimal to hexadecimal
* Help with conversion from binary to hexadecimal
	+ The binary value of 0x AC3 is 0b 1010 1100 0011. The 0x denotes that the characters following have hexadecimal value. Similarly, 0b denotes that the characters following have binary value. (Tip: When converting to binary to hexadecimal remember that one hexadecimal character is equal to 4 binary characters ex. 0xF = 0b1111 (decimal 15), 0xE = 0b1110(decimal 14), 0xD = 0b1101(decimal 13), 0xC = 0b1100(decimal 12),  0xB = 0b1011(decimal 11), 0xA = 0b1010(decimal 10), 0x9 = 0b1001(decimal 9) ... Notice how the binary digits represent the hex digit. You can use this fact to convert hexadecimal more quickly.) For our example we can use this to quickly apply 0xA = 0b1010, append 0xC = 0b1100 and append 0x3 = 0b0011. Giving us 0xAC3 = 0b 1010 1100 0011.
* Call by Value
	+ Does computations in the function, but does not change the value of the variables outside of the function
* Call by Pointer
	+ Changes the value of the variable(s) passed as function parameters
* Call by Reference
	+ Changes the value of the variable(s) passed as function parameters (notation is different from call by pointer
* Review Pointers
* Structs
* New and Delete keyword
	+ Memory leaks, dangling pointers
* Linked Lists
	+ Head and Tail, Next Pointer
* Recursion
	+ Examples will help you understand
	+ Basically, a base case and a recursive case