CSE 109 – Midterm Review

1. Systems programming
    a. What is systems programming?
    b. Why are we interested in it?
    c. How does systems programming compare to applications programming?
2. Operating systems
    a. What is an operating system?
    b. What are some commonly known operating systems?
    c. What resources does an OS manage?
        i. How does an OS manage memory?
        ii. How does an OS manage CPU time?
3. C vs C++
    a. Genesis of the C language and its relationship to Unix
    b. What is bootstrapping?
    c. How is C different from B?
    d. How is C++ different from C?
4. Unix Philosophy
    a. What are the 4 tenants of the Unix Philosophy?
    b. What are the pros and cons of the Unix Philosophy?
    c. What is an alternative to the Unix Philosophy? What are the implications?
    d. What are stdin and stdout? How do they related to the Unix philosophy?
    e. What are Unix pipes?
5. Toolchains
    a. What tools do we use for writing and compiling software?
        i. IDE – visual studio, xcode, Emacs, VIM
        ii. Compiler – gcc, g++
            1. What are the parts of a compiler?
                a. Lexer
                b. Parser
                c. Compiler
    b. What is machine code? How do we generate it from C/C++?
6. Termina
7. ls
    a. What is a terminal?
    b. Why do we still use terminals today?
    c. Terminal tools
        i. ssh
        ii. scp
        iii. ls, cd
8. C and C++ language syntax and semantics
    a. What is the main() function? Why is it special?
    b. What are argc and argv. Where do they come from?

    c. How does C/C++ object code run on the CPU?
    d. How is C/C++ object code stored on the computer?
    e. What is a source file?
    f. What is an executable/object file? How do you generate it?
    g. Language features
        i. While loops
        ii. For loops
        iii. If statements
        iv. Arrays / Pointers
            1. Dereferencing pointers
            2. Pointer offsets
            3. Allocating pointers/arrays
        v. Functions
            1. Passing by value
            2. Passing by reference
9. Stack vs. Heap
    a. What is the stack? What kind of variables are stored on the stack?
    b. What is the heap? What kind of variables are stored on the heap?
    c. How do we declare a stack variable?
    d. How do we declare a heap variable?
    e. When is a stack variable freed?
    f. When is a heap variable freed?
    g. What are the pros and cons of the stack? When would you use it?
    h. What are the pros and cons of the heap? When would you use it?