CS 110
Study Guide for Midterm Exam

Problem solving in Python. Turtles.
Components of a computer
Anatomy of memory
Address, byte, bit
Von Neumann architecture
Role of the operating system
Role of application software
High level language / machine language
Programming language – syntax and semantics
Source file – object file – executable file
How to include a header file and how to define a constant macro
Reserved words, standard identifiers and user-defined identifiers
Role of function main() in a C program
Variable types and declarations
Type hierarchy
Casting and promoting
Know how to create simple printf() and scanf() statements
Arithmetic expressions – operators, precedence, type of an expression
Code reuse
Libraries
Top-down design
Advantages of programming with functions
Function prototypes and function definitions
Function return type
Formal parameters and actual parameters
Pass by value parameter passing
Parameter mapping when a function is invoked
How to invoke a function with and without parameters
Activation records and the stack – know when activation records are created, when they are destroyed, and what is in them
Program stack
Scope – local and global identifiers; notion of a block; be able to identify which identifiers are visible in each block of a program
Be able to create if and if-else statements from a problem description
Know the relational, equality, and Boolean operators
Be able to evaluate a Boolean expression
Know how to create the complement of a Boolean expression
Be able to create a for or while loop given a simple problem description
Given an entry/continuation condition for a loop, be able to give the exit condition
Know the role of the Loop Control Variable, when it is initialized, tested, and updated

Know how to use the pre-/post-increment and decrement operators (++ and —)
Know how to declare, initialize, and use a pointer variable
Know the address-of (&) and dereference (*) operators
Know the difference between pointer and non-pointer types
How much memory does a pointer variable require?
Be able to draw a map of memory showing how pointer and non-pointer variables are laid out in memory
Know what stdout, stdin, and stderr are
Be able to show how you do file I/O redirection from the command line
Be able to write a short piece of code to access a file for reading or writing (fopen, fprintf, fscanf, fclose). *For example*, to read in a triple of integers from a text file ("src.txt") and write their sum to stdout.
Explain the differences between pass-by-copy and pass-by-address
What reasons are there for using pass-by-address parameter passing?
Be able to write a function that uses pass-by-address and to show how it would be invoked
What constraints are there on the actual parameters for pass-by-address that don't exist for pass-by-copy?
Be able to declare an array and show its layout in memory
In what way are the following similar and different?

```
int grades[10];
int *iptr;
```

Provide a for loop that initializes grades to your favorite negative number.
Provide an example that shows the use of an array in a formal parameter list and then in the corresponding actual parameter list.
When passing an array to a function, why do we need to also pass along its size?
Provide a declaration for a C string that is initialized to store your last name.
How many bytes does this string require?
What is the definition of a C string?
Write a function check_len that returns 1 if its two string parameters have the same length; otherwise it returns 0.