# CptS 121 - Program Design and Development

## Exam 1 Review Guide

This document will serve as a guide to help you prepare for the first exam in CptS 121. You will find information about the exam format and topics you are expected to review within this guide.

**What to Bring?**

- Your WSU ID
- Two sharp pencils
- A "cheat sheet" (see below)
- Calculators and other notes may **not** be used during the exam!

**The "Cheat Sheet"**

The exam will be closed-book, but you will be allowed a “cheat sheet": **one side** of a page whose dimensions may not exceed 8-1/2" by 5-1/2" (i.e., one-half of a standard sheet of notebook paper). You must present your cheat sheet to your instructor at check-in, so that he can verify that it meets regulations. If you use a cheat sheet that exceeds the allowable dimensions, or that has writing on both sides of the page, you run the risk of its being confiscated prior to the exam. This policy will be strictly enforced.

**Exam Timeframe**

Please be aware that, because you will be taking the exam during a normal lecture period, **time will be extremely tight** for the exam. *If you show up late to class, you will have less time to take the exam*. Note that, when you hand in your exam, you will be required to present your WSU ID and your cheat sheet to the exam proctor.

**Exam Format**

Expect the exam to look like an hour version of quizzes, with a few more involved problems that are more in the spirit of a lab exercise. Roughly 50% of the exam will be dedicated to concepts, with the other 50% being dedicated to C programming problems. For the concept section of the exam, expect true-false, fill-in-the-blank, multiple-choice, and/or short-answer questions similar to those found on the quizzes. For the programming problem section of the exam, I will present you with programming problems, and you will be expected to write syntactically-correct C code solutions that exercise good design. Note that your solutions do not have to be documented!

**Exam Coverage**

The exam covers the first five weeks of the semester, chapters 2 through 4 of the Hanly and Koffman textbook.

**Chapter 2: Overview of C**

- What is an *algorithm*?
  - We should specify algorithms with *pseudocode* before we implement them in C
- Define what is a variable
  - Memory is allocated when a variable is declared
- Define what is a datatype
- List three types supported by C
  - Integer (int)
  - Double precision floating-point (double)
  - Character (char)
- Define the role of preprocessor directives
- Define and apply escape sequences (the newline (\n) is an example of an escape sequence)
  - Other escape sequences include horizontal tab (\t), backslash (\\), and double quote (\")
- Describe the role of main ( ) in every C program
- Explain the purpose of the { and } curly braces used with main ( ) (and any function for that matter)
  - Every function body must begin with a { and end with a }
- List an example of a literal string
  - "CptS 121 is amazing!"
  - Literal strings are used in printf ( )
- Explain the role of *return 0* in main ( )
  - Indicates the program terminated successfully
- Describe what is a syntax error; which software tool reports these errors?
- Explain what is a logic error
- Describe the rules for naming an identifier; consider C constraints and naming conventions used in the class
- C is case sensitive
- Define and apply Boolean expressions
- Apply operators to C types
  - Arithmetic operators include: +, -, *, /, and %
- Construct and evaluate valid numerical expressions, including mixed-type expressions
- Apply operator precedence
  - *, /, % have the same precedence
  - +, - have the same precedence, but lower than *, /, %
- Distinguish between = and ==
- Compare implicit type casting to explicit type casting
- Apply printf ( ) and scanf ( )
- Describe what is a prompt
- Describe and apply elements of "good" C style and "best" programming practices
- Provide logical memory diagrams that represent variables
- What is the general algorithm applied to problems in this course?
  - Get inputs
  - Perform computations

- Output results

**Chapter 3: Top-Down Design with Functions**

- Apply top-down design principles
  - Divide and conquer
  - Structure charts
- Apply bottom-up design principles
  - We apply bottom-up implementation frequently in the course; we implement solutions to the sub-problems before we implement main ( )
- Define what is a function in C
  - General rule-of-thumb is 1 function = 1 algorithm = 1 task
- What is *cohesion* and *coupling*?
- Construct functions that solve sub-problems
- Define *formal parameter, actual argument, local variable, function header*
- Describe what is a function prototype
  - Declaration to compiler of newly defined function; name, order of arguments and corresponding types, and return type are very important
- Cite advantages of defining functions in C
- What are the 3 parts to a function header?
- What is a function call? How do *actual arguments* relate to function calls?
- Define what is *scope*
- How do functions communicate with main ( ) and vice versa?
- What is a *calling* function? What is a *called* function?
- Apply C math library functions
  - Some of these include: sqrt ( ), pow ( ), floor ( ), ceil ( ), sin ( ), cos ( ), etc.
- Define and apply test drivers
- Arguments in C are passed-by-value (copies)
- What is *functional decomposition* and *abstraction*?

**Chapter 4: Selection Structures**

- What is a conditional statement?
- Describe what is short circuit evaluation?
  - Applies to conditional statements with compound logic
- Construct flowcharts for a particular problem
- List and apply relational operators
  - These include: <, >, <=, >=, ==, !=
- List and apply logical operators
  - These include: !, &&, ||
- Construct if-else statements in C
- Construct nested if statements
- How is *false* defined in C? How about *true*?
- Apply Boolean operator precedence

**File Processing**

- What is a file stream?
- Apply fopen ( ), fclose ( ), fscanf ( ), and fprintf ( )
  - You should be able to open files for read and write modes

- Which standard streams are created when a C program executes?

**Other Topics**

- What is the three file format?
- What is a model? How do they apply to software development?

**Recommended Strategy for Preparing for the Exam**

I recommend that you use the following activities and materials to prepare for the exam:

- **Review quizzes and lab exercises**: These may well be your best resource. An excellent learning activity would be to retake the quizzes and review the lab exercises.
- **Lecture slides and example code**: Study the lecture slides and example code.
- **Read the textbook:** Read or re-read chapters 2 – 4 in your textbook.