CSSE 132 – Introduction to Systems Programming
Rose-Hulman Institute of Technology

Final Exam Review Guide

This exam measures your mastery of these learning objectives:

1. Describe the functions of common computer system hardware elements including CPU, memory hierarchy and input/output devices.
2. Implement and analyze software in the C programming langauge using:
   - Standard C data types
   - Binary arithmetic, boolean and logical operations
   - Functions
   - Arrays
   - C Strings
   - Pointers and Pointer Arithmetic (including Function Pointers)
   - Static and Dynamic memory allocation techniques
   - User and file input/output
   - Command-Line arguments
3. Discuss why certain abilities such as information representation, network communication, input/output, and security require support from multiple layers of a computer system.
4. Design and implement simple IP-based network applications using socket level programming and C.
5. Demonstrate ability to perform tasks like these in a variety of operating environments including the Linux system environment:
   - compile software
   - debug software
   - secure files
   - leverage a version control system
   - manipulate data
   - command-line (shell) navigation and manipulation

1 Format

Two-part exam. 4 hours total time.

- Part 1 (paper). By-hand problems. **Closed resources**. You are allowed to use one **double-sided** 8 1/2 by 11 inch sheet (or two single-sided sheets) of hand-written notes and a calculator. A simplified ARM guide is provided on the last page of the paper part for your reference.
- Part 2 (coding). You are allowed to use only these acceptable resources:
  - Your computer
  - Your assignments and labs submitted in your individual repository for this term
  - The CSSE 132 course website and things directly linked from it

You are not allowed to use other Internet resources, instant messaging, your smartphone, or other communication means during any part of the exam. Use of other resources is considered academic dishonesty and will result in a penalty grade.

2 Topics to study

- Numbers (Objectives 2 and 3)
  - Number representation in binary, hexadecimal, and decimal.
  - Conversions from one number system (binary, hex, decimal) to another.
  - Two's complement
  - Floating point
- Memory (Objectives 1 and 3)
  - Memory addressing, endianness
  - Memory hierarchy: types of memory (DRAM, SRAM, SSD/Flash, Hard Disk)
  - Effective access time (given a cache strategy and hit rate)
- Command Line Interface (Objectives 2 and 5)
  - Access your Pi using SSH
  - Create and edit files. Use Git to obtain, track, and store files and changes.
  - Use basic command line tools (like cd, cat, ls, grep, vim or emacs) to navigate, search and manipulate files.
  - Compiling, running, and debugging Assembly and C programs

- ARM Assembly (Objectives 1, 2, and 3)
  - Reading assembly: Given code, what does it do? Write corrsponding C code
  - Writing assembly: basic instructions, pointers, branches, loops
  - Conventions for procedure call, return address, arguments, return value
- C Programming (Objective 2)
  - Representing data
    - C data types `float`, `int`, `char`, etc.
    - Floating point `float` standard, range and precision
    - C arrays, pointer arithmetic, address (`&`) and dereferencing (`*`)
    - C strings, null-termination, string library functions (`strlen`, `strcpy`, `strcmp`, etc.)
    - C structs: define, allocate, set members, use members
  - Function calls
  - printf and format specifiers
  - Pass by value vs. "pass by reference" using pointers
  - Creating a C program from scratch that can take command-line arguments
- Memory Allocation (Objectives 2 and 3)
  - Allocating and using memory on the stack (in C and ARM assembly)
  - Allocating, using, and freeing memory on the heap (in C)
- Input/Output (Objectives 2 and 3)
  - Similarities and differences between Buffered and Unbuffered IO
  - Getting input from a user in C (especially `fgets`)
  - Opening, closing, seeking, reading, and writing files in C
- Network Programming and Sockets (Objective 3 and 4)
  - Basic network building blocks (hubs, switches, routers, etc)
  - Addressing computers ("hosts") on the network, IP address and port
  - The roles of clients and servers
  - Creating sockets on the client and server

3 Problem-by-Problem Review

Paper Part (60 pts)

1. Problem 1: Multi-choice (10 problems × 2 = 20 pts)
   - 1.1 Socket concept *[Quiz 22/24, Lab7/8]*
   - 1.2 Ponters/Arrays *[Quiz 08]*
   - 1.3 C Programming Basics *[Quiz 07]*
   - 1.4 `git` commands *[used in every coding assignment]*
   - 1.5 `fgets/fread` *[Quiz 19/20]*
   - 1.6 Endianess *[Quiz 03]*
   - 1.7 Compile/Assemble/Link *[Quiz 04]*
   - 1.8 C `struct` *[Quiz 09]*
   - 1.9 Memory Types: SRAM/DRAM/SSD *[Quiz 06]*
2. Problem 2: Number Representations (15 pts)
   - Binary/Hexadecimal/Decimal Signed/Unsigned *[Quiz 02]*
   - Floating Point ☹ *[Q15]*
3. Problem 3: Pointers/`argv` (15 pts) *[Quiz 08/13/25]*
4. Problem 4: Fill in blanks (10 pts) *pointers for output* + `struct` ☹☹ *[Quiz 25]*

Coding Part (90 pts)

1. `part1.s` (20 pts) *[Homework 2/3 Coding]*
   - 1.1 (8 pts) Implement `if/else` also use `str/ldr` to access memory
   - 1.2 (12 pts) Read an `int` array and do some basic analysis *[Quiz 05]*
2. `part1.c` (30 pts)
   - 2.1 (10 pts) `malloc` + `struct` *[like deb_alloc in Lab5]*
   - 2.2 (5 pts) strings and pointers
   - 2.3 (7 + 8 pts) Implement one method to generate a new string ☹
     - Write two functions: `xxxx_len, xxxx`
     - Hint: using `memcpy, strncmp`
3. `part2.c` File I/O (20 pts) *[Exam2 part2]*

- Create a new file to write a program
- Read file(s) and output some results ☹☹ *use the trick of inserting a* `\0` *to make a string*

4. `part3.c`: Socket Programming (20 pts) *[Lab 7/8 Homework 7]*
    - Create a new file to write either a client or a server (the other end will be provided)
    - Implement a certain protocol (ways to communicate between the server and the client)
        - Extract a substring from the message
        - Use `sprintf` to construct a message to send
        - Don't forget to use the return value of `recv()`.