Princeton University
COS 217: Introduction to Programming Systems
Fall 2008 Midterm Exam Preparation

Topics

You are responsible for all material covered in lectures, precepts, assignments, and required readings. This is a non-exhaustive list of topics that were covered:

1. Number Systems

- The binary, octal, and hexadecimal number systems
- Finite representation of integers
- Representation of negative integers
- Binary arithmetic
- Bitwise operators

2. C programming

- The program preparation process: preprocess, compile, assemble, link
- Program structure: multi-file programs using header files
- Process memory layout: text, stack, heap, rodata, data, bss sections
- Data types
- Variable declarations and definitions
- Variable scope, linkage, and duration/extent
- Constants: #define, constant variables, enumerations
- Operators and statements
- Function declarations and definitions
- Pointers; call-by-reference
- Arrays: arrays and pointers, arrays as parameters, strings
- Command-line arguments
- Input/output functions
- Text files
- Structures
- Dynamic memory mgmt.: malloc(), calloc(), realloc(), free()
- Dynamic memory mgmt. errors: dangling pointer, memory leak, double free
- Abstract data types; opaque pointers
- Void pointers
- Function pointers and function callbacks
- Parameterized macros and their dangers (see King Section 14.3)

3. Programming-in-the-Large

- Testing
  - External testing taxonomy: boundary condition, statement, path, stress
  - Internal testing techniques: testing invariants, verifying conservation properties, checking function return values, changing code temporarily, leaving testing code intact
  - General testing strategies: testing incrementally, comparing implementations, automation, bug-driven testing, fault injection
- Debugging heuristics
  - Understand error messages, think before writing, look for familiar bugs, divide and conquer, add more internal tests, display output, use a debugger, focus on recent changes

- Building
    - Automated builds, partial builds
- Performance improvement techniques
    - Execution efficiency: do timing studies, identify hot spots, use a better algorithm or data structure, enable compiler speed optimization, tune the code
    - Space efficiency: use a smaller data type, compute instead of storing, enable compiler size optimization
- Program and programming style
    - Top-down design
- Data structures and algorithms
    - Linked lists, hash tables, memory ownership
- Module qualities:
    - Separates interface and implementation, encapsulates data, manages resources consistently, is consistent, has a minimal interface, reports errors to clients, establishes contracts, has strong cohesion, has weak coupling
- Generics
    - Generic data structures via void pointers, generic algorithms via function pointers, wrappers

4. Applications

- De-commenting
- Lexical analysis via finite state automata
- String manipulation
- Symbol tables, linked lists, hash tables
- Dynamically expanding arrays

5. Tools: The Unix/GNU programming environment
    - Unix, Bash, Emacs, GCC, GDB, Gprof, Make

Readings

As specified by the course "Schedule" web page...

Required:

- *C Programming* (King): 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 22
- *Computer Systems* (Bryant & O'Hallaron): 1

Recommended:

- *Computer Systems* (Bryant & O'Hallaron): 2
- *The Practice of Programming* (Kernighan & Pike): 1, 2, 4, 5, 6, 7, 8
- *Programming with GNU Software* (Loukides & Oram): 1, 2, 3, 4, 6, 7, 8, 9