Programming Languages

1. [General]
   a. Why do we need computer languages?
   b. Explain the differences between compilers and interpreters.
   c. Explain the compile/link/load/execute process.
   d. Explain the steps of a compile (lexical analysis, syntax analysis, etc.)
   e. Explain lexical, syntactic and semantic. Give examples of a lexical rule, syntax rule, etc.

2. [Grammars]
   a. What is the purpose of a BNF grammar?
   b. Explain how operator precedence and operator (left/right) association are handled in BNF?
   c. What is EBNF and how does it differ from BNF?

3. [Binding]
   a. What is binding?
   b. What are some of the possible binding times?
   c. Give an example to support the statement
      "Early binding gives efficiency while late binding tends to give flexibility."

4. [Data Types]
   a. What is a type?
   b. What is strong typing?

5. [Scope and Lifetime]
   a. What is a scope rule?
   b. How does the scope rule of C++ differ from that of Ada?
   c. What is a storage class?
   d. What are the possible storage classes in C++?
   e. Explain the differences between scope and lifetime.

6. [Parameters]
   a. What are the mechanisms for parameter passing (value, reference, value-result, name)?
   b. Give an example which different parameter passing schemes may yield different results.

7. [Run-time Environment]
   a. What is a call stack?
   b. What is an activation record?
   c. What is generally stored in an activation record?
d. Explain the functions of static links and dynamic links.

e. Explain the steps involved in a procedure call using activation records and the call stack.

8. [Package]
a. What is an Ada package?
b. Compare and contrast Ada packages and C header files.
c. How is a package different from a class?

9. [Encapsulation]
a. What is the purpose of an encapsulation?
b. How are encapsulation achieved in Ada, C, and C++?

10. [Generic/Template]
a. What is a generic program unit?
b. What is the purpose of generic units?
c. Why is necessary that a generic unit be instantiated before it's used?

11. [Re-usability]
a. Give five ways that software components can be modified for reuse. [Extension, restriction, redefinition, abstraction, and polymorphism]
b. Explain how C++ incorporated these schemes.
c. Explain the difference between overloading and polymorphism.
d. Explain why polymorphism requires dynamic binding.
Data Structures

It is important to realize that it is assumed you can respond accurately to specific questions described in the text or class. You must answer such questions correctly. In general you will be judged by how well you can extrapolate these ideas to situations not explicitly covered in the text or class. You should expect to demonstrate the algorithms at the black board with live data.

1. Explain in detail how sorting is accomplished in auxiliary storage using the polyphase sort. Discuss other external sorting algorithms covered in the text such as replacement selection, cascade, and Shell's methods.
2. Discuss radix sorting, shell sort, quick sort, bubble sort, sort by selection, sort by hashing, heap sort, sort by counting, topological sorting, etceteras in detail. Compare with respect to time and space required. Explain data dependencies.
3. When is linked list technology superior to sequentially allocated list technology and vice versa.
5. Discuss "garbage collection" in detail. Explain specific algorithms for dynamic storage allocation as well as reclamation. For example, how would you implement the malloc operator in "C" or "new" operator in C++. Apply your discussion to both simple data types such as integers as well as objects.
6. Discuss balanced tree insertion algorithms.
7. Suggest an optimal sorting algorithm for a parallel computer and explain why it is optimal.
8. Explain the use of priority queues. How are they implemented?
9. Derive formulas for the average number of probes to find an item in a random sequential list or sorted sequential list. Other simple derivations or explanations should be expected. For example why is the approximate number of probes to find an item in a sorted sequential list using a binary search log base 2 of N where N is the number of items in the list?
10. Demonstrate at least two methods for implementing a binary search. Under what circumstances is each best? How much better?
11. What is a Trie? How and why would you use one?
1. Discuss coupling and cohesion in detail. How do they relate to control flow design and programming. Explain how they relate to object oriented design and programming.
2. What is Software, what is software engineering? How to evaluate software quality? How to achieve the quality?
3. How do programmers spend their time?
4. Discuss the factors that affect software quality and productivity.
5. Discuss software process models such as the prototyping model, the spiral model, the incremental model, the component model, Agile model, …
6. Discuss the cost estimation techniques. Explain COCOMO model in detail. How to perform staff level (personnel) estimation?
7. Discuss Function Points and related technology with respect to the management of software projects.
8. What is the purpose of the requirement engineering? How to obtain and capture the requirement?
9. What kind of risks a software company may take? How to analyze and manage the risks?
10. Give examples to show how to use the following techniques on the requirement specification: implicit equations, recurrence relations, algebraic axioms, regular expression, decision tables, event tables, transition tables, and Petri nets.
11. Explain the following fundamental design concepts: abstraction, structure, information hiding, modularity, verification, and aesthetics.
12. Be able to use UML notations for system design: Data flow diagram, Structure charts, structured flowcharts, Use cases, State Machine, and other object oriented notations.
13. How to manage a project to make sure it completes on time?
14. Object oriented analysis: use examples to explain classes and objects, generalization -specialization relationship and whole - part relationship.
15. Explain what the polymorphism is.
16. Use examples to show good coding style and bad coding style.
17. Discuss management organizations meant to support team development (chief programmer teams, democratic team, hierarchical team, matrix structure)
18. Explain software re-engineering and reuse (specifications, designs, and code). How can this be accomplished in an industrial environment?
19. Discuss software metrics in detail. What should be measured and how? How should the statistics obtained through measurement be applied to improve the software process? What is a software process?
20. Discuss future trends in our field.
21. Discuss the importance of user's interface and guidelines in interface design.
22. Discuss software testing techniques. What is black box testing? What is white box testing? What is statement covering? Branch covering? And logical path covering?
23. Use an example to explain the input/output assertion.
Operating Systems I

Suggested References (Any one of these):


Introduction
1. What actions does an operating system perform?
2. What services does an operating system provide?
3. What forces have driven the development of operating systems?
4. List several major milestones in the development of operating systems.
5. Define "multi-processing" and "multi-programming", being careful to explain the difference between them.
6. Explain distributed systems.

Computer System Structures
7. Discuss the basic architectural components of a modern computing system, such as cache, registers, ALU, CPU, system bus, primary memory, device controllers, and hard disks. How are these elements typically structured and organized. How do they interoperate?
8. Explain interrupts and how the functioning of a modern computing system depends on interrupts.
9. Explain DMA.
10. Explain privileged mode and privileged instructions.
11. Discuss the way that the existence of two operating modes facilitates the maintenance of correct operation in a multiprogramming operating system. Be sure to include a specific example to illustrate each important point you make.
12. Discuss some important operating systems capabilities and the hardware support required to implement them.
13. Explain system calls.

Operating System Structures
14. An operating system can be thought of as a supplier of many services to user processes, and a manager of many resources that are available to user processes. Name some of these services and resources
15. What is a system call? Who (what type of process) would MAKE a system call? Who (what process) would be the CALLEE?
16. Explain what a process is.
17. What are the services that an operating system provides in relation to:
   a. Process Management
   b. Main-Memory Management
   c. Secondary Storage Management
18. Discuss several approaches to designing an operating system.

Processes and Threads
19. Define "program" and "process", being careful to explain the difference between them.
20. What is the "state" of a process? List some of the components of a process state.
22. Explain process control blocks.
23. Explain ready queue and I/O queue.
24. Explain the roles of short term and long term schedulers.
25. Explain swapping and context switching.
26. Discuss some important Unix system calls.
27. Explain user-level versus kernel-level threads.
28. Explain the producer-consumer problem.
29. Discuss several styles of interprocess communication.

CPU Scheduling
30. Explain preemptive and non-preemptive scheduling.
31. Explain shortest-job-first (SJF) scheduling. What does it achieve? When is it easy to implement, and when is it hard to implement?
32. Explain "aging" and what the need for it is.
33. Explain round robin.
34. Explain symmetric multiprocessing.
35. Explain throughput, turnaround time, waiting time and response time.

Process Synchronization
36. Define "critical section" and "critical section problem". Be sure to adequately describe the context in which these concepts are important.
37. Explain the meaning of each of the following three requirements for a solution to a critical section problem: a) mutual exclusion b) progress c) bounded waiting.
38. What is a "software solution" to a critical section problem? When we studied software solutions, what underlying assumptions did we rely upon concerning the instruction set of the machine on which our programs would execute?
39. Explain how the "hardware" of a computer can be used to insure the atomicity of an operation such as the "wait" of a semaphore, or a "test-and-set".
40. Give a complete description of one of the following "classic problems" in process coordination: a. The producer/consumer problem (bounded buffer version) b. The readers/writers problem c. The dining philosophers problem.
41. Define "busy waiting". How can busy waiting be avoided in the synchronization protocols of a uni-processor computer?
42. Describe the data structure "counting semaphore" (also called a "queueing semaphore"). Be sure to describe the data object itself, its structure and organization from a logical standpoint, the operations that are available, etc.
Deadlocks
43. What are the four necessary conditions for deadlock? Discuss some of the (operating system) techniques that prevent deadlock by making sure that one or more of the conditions never holds. What are some of the details and requirements of the implementation of these techniques?
44. Discuss deadlock avoidance. What is involved in the Banker's algorithm? How does an operating system use the Banker's algorithm to avoid deadlock? What is a safe sequence? If a system is not in deadlock, is it always in a safe state?
45. Describe a method by which an operating system can detect deadlock. How should this be done in a system in which there is just one instance of each resource type? What about a system in which there are multiple instances of some resource types?

Memory Management
46. Discuss various approaches to the management of main memory. Include fixed partitions, variable partitions, segments, and paging.
47. What are external and internal fragmentation?

Virtual Memory
48. Discuss virtual memory, and various approaches to the implementation of virtual memory.
49. Discuss implementations of page replacement.
50. Discuss the issues involved in per-process frame allocation.
51. Discuss ways of detecting and dealing with thrashing processes.
52. Discuss the issues involved in choosing a page size.

File-System Interface
53. Describe several approaches to organizing a file system. Explain advantages and disadvantages of the various methods.
54. Discuss some of the problems inherent in maintaining a directory structure in which arbitrary multiple links to files are allowed to exist.

File-System Implementation
55. How should blocks on magnetic disk be numbered so that allocation of a series of blocks will result in a minimal number of seeks for sequential accesses to files?
56. Discuss the design goals important in the choice of a data structure for representing the free space list for a magnetic disk.
57. Define contiguous, linked, and indexed file allocation methods. What kinds of file accesses are supported efficiently by each? What are some advantages and disadvantages of each? Which method suffers from the effects of external fragmentation?
58. In order to support create, open, read, write, reset (seek), and delete operations on files, what information about each file must be stored by the operating system?

I/O Systems
59. Discuss the ways that typical device controllers interact with the CPU and other components of a modern computing system.
60. Discuss polling, interrupt driven I/O, and DMA.
61. Discuss interrupt levels.
62. Discuss block-oriented and character-oriented I/O.

Secondary Storage Structure
63. Explain various disk scheduling policies.
64. What is involved in disk formatting?
65. Discuss swap space implementations.
66. Discuss disk striping and RAID's.

Network Structures
67. Discuss various approaches to managing and coordinating multiprocessor systems. Be sure to include loosely coupled vs. tightly coupled, and master/slave organization.
68. Describe various network topologies.
69. Define "Distributed System”.
70. What are the advantages of a distributed system?
71. Describe several popular connection topologies of distributed systems.

Distributed-System Structures
72. Discuss computation and process migration.
73. Discuss fault-tolerance.
74. Discuss scalability.

Distributed-File Systems
75. Discuss processor and storage device transparency.
76. Discuss the relative merits of several distributed file systems.

Distributed Coordination
77. Why do we require a process to advance its logical clock when it receives a message whose timestamp is greater than the current value of its logical clock?
78. Discuss ways that distribute processes can achieve mutual exclusion.
79. Discuss ways that distributed processes can achieve atomic distributed transactions.
80. Discuss ways that distributed processes can prevent deadlock.
81. Discuss distributed deadlock detection.
82. Discuss election algorithms.

Protection
83. Discuss domains, capabilities, access lists, and access matrices.

Security
84. Discuss authentication procedures.
85. Discuss threats such as trojan horse, trap door, worm, virus.
86. Discuss threat monitoring.

The Unix System
87. Discuss Unix design principles and innovations.
88. Discuss fork, execve, wait, and vfork.
89. Discuss CPU scheduling under Unix.
90. Discuss the 4.2BSD file-system.
91. Discuss the advantages and disadvantages of writing the majority of an operating system in a high-level language.

Operating Systems II

1. OS evolution
   a. Terms including Monitor, Batch Processing, Uniprogramming OS, Multiprogramming OS, Timesharing OS, multiprocessing system, SMP,
   b. Concept of processes, memory management, virtual memory protection and security, scheduling and resource management
2. Basic Machine Architecture at the Computer Organization Level
   a. Please explain how memory indirect (indexed, immediate) addressing works.
   b. Please explain low level interrupt processing mechanisms - what is an interrupt handler (device driver) and how is it called
3. Processes and Threads
   a. Idea of process states and process state diagrams: ready, running, blocked (I/O), etc.
   b. Idea of process context - what does the context consist of? According to Stallings, what is the difference between a process switch and a context switch - this is a good distinction
   c. PCB – Process Control Block
   d. Idea of multithreaded processes
      i. Why is multithreading important?
      ii. Why is an OS that provides for multithreading better than a “normal” multiprogrammed OS?
      iii. What does SMP mean?
      iv. Why is a multithreaded OS important given the fact that many new computer systems will be of the SMP variety?
   e. TCB – Thread Control Block
4. Scheduling Algorithms (for use by the dispatcher)
   a. Explain criteria used in scheduling – response time, turnaround time, throughput, deadlines, device, utilization, predictability, fairness, enforcing priorities – deadlines?
   c. What dose preemptive and non-preemptive mean?
   d. Multiprocessor or thread scheduling algorithms – what are they and how do they work?
      1. Load sharing, gang scheduling, dedicated processor assignment, and dynamic scheduling – how do they work and why use them?
   e. What is real – time scheduling and how does it work?
5. Coordinating Cooperating Sequential Processes
   a. What does mutual exclusion enforcement mean?
   b. What is a critical section?
   c. What do Bakery algorithms do? What OS system support do they require? (shared memory)
      How do they work?
   d. Mutual Exclusion Enforcement mechanisms
      i. Software algorithms – adv and disadv? (busy wait?)
      ii. Hardware instructions – (test and set and exchange) How do they allow us to enforce mutual exclusion? Adv and disadv (busy wait?)
      iii. OS mechanisms - semaphores
         a. Primitives are P and V, up and down, or wait and signal (from Stallings)
         b. Binary and Counting (according to Stallings) – how are they implemented, how can you use them to enforce mutual exclusion, how can you use them to solve other problems beyond mutual exclusion
         c. Producer consumer both infinite buffer and infinite buffer
         d. Readers and writers – readers have precedence and writers have precedence
e. Dining philosophers problem

iv. Language Mechanisms – monitors, protected records
   1. How do you use them to enforce mutual exclusion
   2. How do you use them to control processes in a method similar to semaphore functionality
   3. Java and Ada support

v. Explain the details of message passing.
   a. What are the primitives? (Send and receive)
   b. How do you use message passing to enforce mutual exclusion
   c. Explain the difference between the various strategies of message passing such as blocking send/blocking receive or non-blocking send but a blocking receive, etc
   d. What are sockets and how do they expand message passing facilities?

e. Problems of deadlock and starvation
   i. What is deadlock and what is starvation
   ii. What are the necessary conditions for deadlock in an OS?
   iii. What are the standard OS mechanisms to handle deadlock?
      a. Deadlock prevention
      b. Deadlock avoidance
      c. Deadlock detection

6. Memory management
   a. Partitioning of memory
      i. Fixed size partitions
      ii. Variable sized partitions: First-Fit, Next-Fit, Best-Fit, Worst-Fit. How do they work and compare adv and disadv?
      iii. What is Virtual memory and how does it work?
         a. Adv and disadv?
         b. You should know and be able to explain how paged and segmented VM works
         c. What is a page table or segment table and how do they differ?
         d. What about combination segmented/paged virtual memory systems?
         e. What is a TLB and how does it work?
         f. What is a page fault and what does thrashing mean and how can you cause thrashing?
      g. Consider paged virtual memory
         ii. Resident set size management – fixed or variable
         iii. Page replacement policies – global or local – demand paging or prepaging
         iv. Page replacement algorithms – LRU, FIFO, optimal, clock (how do they work and which are widely used and why?)
         v. Dennings working set model – variable allocation local scope – how does it work? What is delta t? Explain working set estimation algorithms?

7. IO management and disk scheduling
   a. Explain the various disk scheduling algorithms including random scheduling, priority scheduling, LIFO, Shortest Service time first, SCAN, CSCAN, NSCAN

8. Unix Concepts
   a. Idea of a kernel/microkernel
   b. System Calls, System Programs, Command-Shell
   c. Creating processes
   d. File systems using I-nodes. How does the directory structure work, how do you specify which blocks (sectors) belong to a file within an I-node, what is the superblock and how does it work?

8. Networking and Distributed Processing
   a. The idea of a communications architecture that must be the foundation for any network or distributed OS. In particular TCP/IP and OSI.
b. Bully-algorithm

c. What is a network OS and how does it work?

d. What is a distributed OS and how does it work?
   i. What does the term process migration mean?

e. What does Client/Server Computing mean and how would you write a Client Server app?
Database

1. Familiarity is required with all Database jargon and issues. A good introduction to this information is given in the textbook Rob and Coronel - Database Systems Design Implementation, and Management in Chapter I - see the key terms. Also check Date Edition C), Chapters I and 2

2. Familiarity is required with all relational database jargon and issues:
   Relations - Tables or “Files”
   Base Relations and Views
   Tuples - Rows or “records” (number of tuples in a relation-cardinality)
   Attributes - Columns or -fields” (number of attributes is call the degree)
   Domain of an attribute?
   First Normal form for a relation
   Normalization of a relation
   Keys
   Superkeys - can have superfluous attributes
   Minimal Superkeys - no superfluous attributes (other books call these ‘Keys’)
   Primary Key
   Secondary Key
   Candidate Key
   Composite Key
   Foreign Key
   Referential integrity?
   Should know and understand NULL handling.

3. Familiarity with the Relational Algebra is required.
   You should know the language and be able to construct queries in ffic algebra - check examples in Date Ed 6.

4. Familiarity with the Tuple calculus is required
   You should know and understand the simple calculus and be able to construct queries – important understand the use of the quantifiers (existential and universal)

5. You should be able to write queries in SQL.

6. You should know how to design a database using ER modeling.

7. You should understand 1st, 2nd, 3rd, BC, 4th, 5th (Project Join), and DK normal forms. You should know about functional dependencies and multivalued dependencies and join dependencies.

8. You should understand the anomalies that exist that motivate us to normalize our relations to higher normal forms.

9. You should understand the normalization process.

10. You should understand the issues involved with data recovery, concurrency, security, and integrity controls in the database system.

11. With data recovery, you should understand transaction management - commit and rollback

12. With concurrency control, you should understand the problems that cause the requirements for concurrency control including the lost update problem the
uncommitted dependency problem, and the inconsistent analysis problem - The serializable issue.
Locking techniques that support concurrency control
13. Security and Integrity - distinguish between Security and Integrity issues
Understand what encryption is (DES and public key)
Network and Data Communications

1. Differentiate between analog and digital data/signal/transmission,

2. Given $S(t) = 0.5 \sin(2000\pi t + \phi)$
   a. What is the peak amplitude? The frequency? Period?
   b. Let $\phi = 0$, draw the wave.
   c. Let $\phi = \pi / 2$, draw the wave
   d. Let $\phi = 0$, and replace 2000 with 4000, draw the wave.

3. Use an example to explain frequency, spectrum, and bandwidth. Tell what frequency ranges of the following guided media are about: twisted pair, coaxial cable, optical fiber?

4. Transmission impairments:
   a. Explain attenuation and attenuation distortion;
   b. Explain delay distortion
   c. Explain four categories of noises: Thermal noise, intermodulation noise, crosstalk, and impulse noise.

5. Wireless transmission: Indicate some significant differences between broadcast radio and microwave (e.g. the ranges of frequencies, directional and omnidirectional transmission).

6. What is the advantage of a parabolic reflective antenna?

7. Encoding techniques – digital data, digital signals
   a. Understand and compare among the following encoding formats: NRZ-L, NRZI, Bipolar-AMI, Differential Manchester and Bipolar with 8-zeros substitution (B8ZS).
   b. For the bit stream 1100001000000001, sketch the waveform for each of the above formats.
   c. What is differential encoding? Pick one encoding format above to explain the benefit of differential encoding.
   d. Differentiate between data rate and modulation rate.

8. Encoding techniques - digital data, analog signals.
   a. Understand how ASK (amplitude shift keying), FSK (frequency shift keying), and PSK (Phase shift keying) work.
   b. Use an example to show how to use analog signals to transmit digital data.

9. Encoding techniques – analog data, digital signals
   a. Pulse code modulation (PCM) is based on the sampling theorem. What is that theorem?
   b. Use an example to show how to use PCM technique to convert analog data to digital signals.
   c. Use an example to show how to use Delta modulation (DM) technique to convert analog data to digital signals.

10. Explain “Modulo 2 arithmetic”. Perform the following modulo 2 calculations: $1010 + 1111, 1010 - 1111, 1010 \times 1111, 101011/11$

11. What is a parity bit? How parity check works to detect errors.
12. What is the CRC stand for? How it works? Four versions of pattern are widely used, what are they (CRC-12, CRC-ANSI, CRC-CCITT, and IEEE-802).
13. Flow control: Explain how sliding-window is used for flow control.
14. HDLC gives a typical frame structure, give its frame format. What is the pattern of its Flag field? Why “bit stuffing”?
15. What is multiplexing? What do FDM and TDM stand for? Use real applications as examples to explain how each of them works.
16. What is the main difference between circuit-switching and packet-switching?
17. Show a network that is blocking; show a network that is non-blocking.
18. Explain the difference between datagram and virtual circuit operation.
19. What is the significance of packet size in a packet-switching network?
20. Routing
   a. What is the purpose for routing?
   b. Explain each of the following routing strategies: Fixed routing, flooding, random routing, and adaptive routing.
   c. Given a network, how to build a routing table for each node in the network if the distributed fixed routing is adopted?
   d. Explain how the Isolated adaptive routing algorithm works.
   e. What is the purpose of Dijkstra’s least-cost algorithm?
21. LAN (Local area networks)
   a. List four common LAN topologies and briefly describe their methods of operation (bus, ring, tree, star). Which topology is the most popular one?
   b. What functions are performed by a bridge? Use an example to show how several LANs are connected using bridges.
22. There are five layers in TCP/IP protocol architecture. List and explain all of them.
23. IP
   a. Draw IP header (pick a version by yourself).
   b. In IPv4, how many bits are used for a source or destination address? There are three principal network classes, Class A, B, and C. What is the format of each class? For example, given the address 192.228.17.57, which class of the network this node belongs to?
   c. What is the function of the masks? What is the default Class A mask?
24. TCP: Briefly describe the credit scheme used by TCP for flow control.
25. Briefly describe how a programming language supports the implementation of communications.