Topic-wise distribution (100 questions)

Area	Approx. Qs	Examples
Computer Organization & Architecture	~20	number systems, memory hierarchy, cache, addressing modes, word length
Data Structures & Algorithms	~20	stacks/queues/trees, sorting, traversals, complexity analysis, recursion
Computer Networks	~18	OSI/TCP-IP layers, protocols, IP addressing, CSMA/CD, HTTP/HTTPS
Operating Systems	~10	scheduling, virtual memory, semaphores, system calls, paging
DBMS/SQL	~5	SQL queries, aggregate functions, subqueries
Web Technologies	~8	HTML5, JavaScript, AJAX, HTTP methods/status codes
OOP Concepts	~6	polymorphism, inheritance, encapsulation, abstraction
Software Engineering	~8	SDLC, testing types, models (Waterfall/Spiral/RAD), SRS
C Programming (code-tracing)	~5	array manipulation, loops, operator precedence
Cybersecurity	~4	firewalls, phishing, malware types, biometric security

Key observations

1. **Core CS fundamentals dominate** (~58%): computer organization, DS&A, and networks together form the backbone of this paper
2. **Numerical/computational questions are frequent** — memory access time, subnet calculations, Shannon’s capacity, postfix evaluation, memory chip calculations.
3. **Code-tracing questions (C programs)** appear as a distinct recurring block (~5 questions) — testing practical understanding of loops, arrays, and operator precedence rather than just theory.
4. **Networking is heavily weighted and detailed** — going beyond basic definitions into protocol-level specifics (OSPF algorithm, CSMA/CD behavior, addressing modes, subnetting)
5. **OS and DBMS are present but lighter** — likely secondary topics relative to

networks/DS&A/architecture.

6. **Software engineering and OOP are conceptual/definitional** — mostly single-fact recall (SDLC models, testing sequence, OOP pillars), lower difficulty than the numerical sections.
7. **Cybersecurity is a small but modern addition** — phishing, malware types, firewalls — reflecting current-affairs-adjacent inclusion in technical exams.

Takeaway for prep: This paper skews toward applied/computational CS fundamentals — prioritize numerical problem-solving (memory calculations, complexity analysis, networking math) alongside protocol/layer-level networking knowledge and C code-tracing practice, as these together make up the toughest and most differentiating ~50% of the paper.