

All Subjects

Quick Revision Sheet

13 Subjects | Key points, Formulas, Tables | Hindi + English

1	Operating System (OS)	Page 2
2	Database Management System (DBMS)	Page 3
3	Computer Networks (CN)	Page 4
4	Software Engineering (SE)	Page 5
5	Object Oriented Programming (C++)	Page 6
6	Data Structure (DS)	Page 7
7	Theory of Computation (ToC)	Page 8
8	Computer Organization & Architecture (COA)	Page 9
9	Artificial Intelligence (AI)	Page 10
10	Web-Based Application Development (WAD)	Page 11
11	Multimedia	Page 12
12	Internet of Things (IoT)	Page 13
13	E-Commerce	Page 14

1. Operating System (OS)

Basics

- **OS** = interface between user & hardware; manages resources. **Shell** = outer layer, command interpreter. **Kernel** = inner core (Process, Memory, Device, File mgmt).

User → Shell → Kernel → Hardware

- **SUI** (1 user: DOS, Windows, Android, iOS) vs **MUI** (many users: UNIX, Linux, Win Server). ! Android/iOS = Single user, Multitasking.
- **CLI** (text, fast, less memory: DOS, UNIX) vs **GUI** (icons, mouse; needs **WIMP** = Window, Icon, Menu, Pointer).

Gen	Period	Type
1st	1940-55	No OS (vacuum tube, ENIAC)
2nd	1955-65	Batch OS (transistor, IBM 7094)
3rd	1965-80	Multiprogramming, Time sharing (IC, UNIX)
4th	1980-2000	PC OS, GUI (MS-DOS, Windows)
5th	2000-now	Modern: mobile, cloud, AI, IoT, multicore

Types of OS

- **Batch**: similar jobs in a batch, no user interaction. **Time Sharing**: CPU split in time slices (quantum); context-switch overhead.
- **Multiprogramming**: many programs in RAM, CPU never idle (I/O wait → next program). **Multitasking**: time sharing based, user interaction.
- **Multiprocessing**: many CPUs, true parallel. **SMP** = equal CPUs (Linux, Win server); **AMP** = Master-Slave (embedded).
- **RTOS**: deadline mandatory, deterministic. **Hard** (pacemaker, missile, airbag, nuclear) = delay fatal; **Soft** (video streaming, gaming) = only quality loss.
- **Distributed**: many computers look like ONE system, high transparency, fault tolerant (Amoeba, Plan 9, LOCUS). **Network OS**: server-client, each PC has own OS (Win Server, Novell).
- **Embedded OS**: small, low power, fixed task (washing machine, TV, car ECU).

9 Functions of OS

- Process mgmt (scheduler) • File mgmt (FAT, NTFS, APFS, HFS, exFAT, MFS, ZFS) • Memory mgmt • Security (password ≥ 8 chars: A-Z, a-z, 0-9, @#\$%&) • I/O (device drivers) • Integrated • Command (shell: dir, print) • Disk • Error mgmt.

Fit	Search	Wastage
First Fit	from start, first big enough	Medium, fast
Next Fit	from last allocated block	Medium, faster
Best Fit	smallest suitable	Least, slow
Worst Fit	largest block	Highest

CPU Scheduling

TAT = Finish - Arrival | WT = TAT - Burst | Start = previous Finish

Algo	Type	Key point
FCFS	Non-preemptive	arrival order; tie → smaller PID
SJF	Non-pre / Pre	min avg waiting; long job starves
SRTF	Preemptive	least remaining time first
Priority	Pre / Non-pre	lower no. = higher priority; starvation, priority inversion
Round Robin	Preemptive	time quantum; NO starvation; TQ small → more switching, TQ large → FCFS
MLQ	Static	fixed queues, no movement, lower queue starves
MLFQ	Dynamic	process moves between queues, less starvation
HRRN	Non-preemptive	RR = (WT+BT)/BT, highest first
FSS	User based	CPU shared per user/group

- **Starvation** → solution **Aging** (raise priority of waiting process), fair scheduling.

EXAM TIP Solved results: FCFS AWT 3.5 / TAT 6.25 • SJF AWT 2.33 • SRTF AWT 3.5 • Priority AWT 5.25 • RR (TQ 2) AWT 6.

Sync, Memory, Interrupts

- **Deadlock**: processes wait forever for each other's resources. **Race condition**: result depends on execution order. **Critical section**: shared-data code, 1 process at a time.
- **Mutex** = binary lock (0/1). **Semaphore**: Binary (0/1) / Counting (0 to n).
- **Internal frag** = waste inside allocated block (8KB block, 6KB process → 2KB). **External frag** = free memory scattered (10+20+15=45KB, can't fit 30KB).

Total IF = Σ (Allocated - Used)

- **Paging**: page = frame size, no external frag, internal frag in last page. 14KB/4KB = 3.5 → **4 pages**, IF = 2KB. Logical 10KB, page 4KB → **page 2, offset 2KB**. **TLB** = fast cache of page→frame. **Segmentation**: logical parts (code, data, stack, heap), external frag possible.
- **Virtual memory** = disk as RAM extension. **Swapping** = whole process RAM↔swap space. **Thrashing** = only swapping, CPU use very low.
- **Interrupt**: Hardware (keyboard, disk) • Software (system call, divide by 0) • Maskable (can ignore) • Non-maskable NMI (power failure).

2. Database Management System (DBMS)

Basics

- **Data** = raw facts. **Information** = processed data. **Database** = organized collection. **DBMS** = software to store, manage, secure data (MySQL, Oracle, SQL Server).

Feature	File System	DBMS
Redundancy	High	Low
Concurrency	No	Multi-user
Security	Poor	Roles, permissions
Backup	Manual	Automatic
Access	Complex	Easy via SQL

- **Components:** Hardware, Software, Data, Users (DBA, programmers, end users), Procedures.
- **3-Level Architecture:** External (user view) → Conceptual (logical) → Internal (physical).
- **Data independence:** Logical (change conceptual schema, views unaffected) / Physical (change storage, conceptual unaffected). **Schema** = design, **Instance** = data at a moment.
- **OLTP** = daily transactions, current data, fast (banking). **OLAP** = analysis, historical data (data warehouse).

Types of Database

- **Relational:** tables + keys + SQL. **OODBMS:** objects (db4o).
- **Hierarchical:** tree, 1 parent (IBM IMS). **Network:** graph, many-to-many (IDMS).
- **NoSQL** (non-relational, unstructured, big data): Document (MongoDB) • Key-Value (Redis) • Column (Cassandra) • Graph (Neo4j).
- **Distributed** = many locations, looks like one (Spanner). **Centralized** = one server.

ER Model

- **Entity** (rectangle) • **Attribute** (oval): Simple, Composite, Single-valued, Multi-valued, Derived, Key • **Relationship** (diamond).
- **Degree:** Unary (1) • Binary (2, most common) • Ternary (3) • N-ary (>3). Degree of relation = number of columns.

Cardinality	Example	Table rule
1:1	Person-Passport	FK either side
1:N	Dept-Employees	FK on N side
M:N	Student-Course	new bridge table

- **Strong entity** has own PK; **Weak entity** needs a strong entity (partial key + strong PK).
- **ER → Tables:** Entity→Table, Attribute→Column, Key→PK, Relationship→FK. **Generalization** = combine into higher entity (Car, Bus → Vehicle); **Specialization** = split (Employee → Manager, Clerk); **Aggregation** = relationship treated as entity.

Keys

Key	Meaning
Super	any attribute set that is unique (extra attributes allowed)
Candidate	minimal super key (unique + minimal)
Primary	chosen candidate, only ONE, NOT NULL
Alternate	candidates not chosen as PK
Foreign	refers PK of another table; duplicates & NULL allowed
Composite	2+ attributes together unique
Unique	unique, one NULL allowed, many per table
Secondary	for searching only, not unique

Normalization

- **Anomalies** (redundancy): Update (change many rows), Insert (can't add instructor w/o student), Delete (deleting student loses instructor info).
- **FD** $A \rightarrow B$: same $A \Rightarrow$ same B .

NF	Rule
1NF	atomic values, no multivalued/repeating group
2NF	1NF + no partial dependency
3NF	2NF + no transitive dependency ($A \rightarrow B \rightarrow C$)
BCNF	every FD $X \rightarrow Y$ has X a super key

- **Codd's 12 rules** (0-12): relational, info as tables, guaranteed access, null handling, catalog, SQL sublanguage, view update, set operations, physical & logical independence, integrity independence, distribution independence, nonsubversion.

SQL

Type	Commands
DDL	CREATE, ALTER, DROP, TRUNCATE, RENAME (structure)
DML	INSERT, UPDATE, DELETE (data)
DQL	SELECT
DCL	GRANT, REVOKE
TCL	COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION

- **DROP** = structure + data gone; **TRUNCATE** = data gone, structure stays (faster). UPDATE/DELETE without WHERE affects all rows.
- **Constraints:** NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT.
- **LIKE:** % = many chars, _ = one char. **IN** = short OR. **Clauses:** WHERE, GROUP BY, HAVING (on groups), ORDER BY (ASC default), DISTINCT.
- **Aggregates** (marks 80,70,90,60,70): COUNT=5, SUM=370, AVG=74, MAX=90, MIN=60.
- **Joins:** INNER (match only) • LEFT (all left) • RIGHT (all right) • FULL (all both) • SELF • CROSS (cartesian).
- **ACID:** Atomicity (all or nothing) • Consistency • Isolation • Durability (after COMMIT permanent). **View** = virtual table (stores query, not data). **Index** = speeds SELECT, not INSERT/UPDATE.

3. Computer Networks (CN)

Basics

- **5 components:** Message, Sender, Receiver, Medium, Protocol. **Effective:** Delivery, Accuracy, Timeliness, Jitter.
- **Simplex** (one way: radio/TV) • **Half duplex** (turn by turn: walkie-talkie) • **Full duplex** (both together: phone call).

Topology	Pros	Cons
Bus	cheap, simple	cable break = network down
Star	one fail OK, easy	hub fail = all down
Ring	stable	one fail affects all
Mesh	very reliable, $n(n-1)/2$ links	costly, complex
Hybrid	flexible, scalable	complex

Network	Area	Note
PAN	1-10 m	Bluetooth
LAN	building	private, fast
MAN	city 10-50 km	cable TV
WAN	country/world	Internet

- CAN = campus; SAN = storage network; VPN = secure private connection over Internet.

Transmission

- **Parallel:** many bits together, many wires, short distance, fast (CPU↔RAM). **Serial:** 1 bit at a time, 1 wire, long distance (USB, Ethernet).
- **Guided:** Twisted pair (UTP/STP, LAN/phone), Coaxial (cable TV, CCTV), **Optical fiber** (light, fastest, secure, no EMI).
- **Unguided:** Radio (omni, Wi-Fi) • Microwave (point-to-point) • Infrared (short, no walls) • Satellite (global).

OSI & TCP/IP

L	OSI Layer	Work / Example
7	Application	user services: HTTP, SMTP, FTP
6	Presentation	format, encrypt, compress: JPEG, TLS
5	Session	establish/maintain/end session, sync, token
4	Transport	end-to-end, error & flow ctrl: TCP, UDP
3	Network	IP address, routing: Router
2	Data Link	MAC, frames, error detect: Switch
1	Physical	bits, cables: Hub

- **TCP/IP 4 layers:** Application (OSI 7+6+5), Transport, Internet, Network Access (OSI 2+1). OSI = theoretical (ISO); TCP/IP = practical (DARPA).

Device	Layer	Function
Hub / Repeater	1	broadcast / boost signal
Switch / Bridge / NIC / AP	2	MAC based forwarding
Router	3	IP based routing, LAN↔WAN
Gateway	all	protocol conversion
Modem	1	digital ↔ analog
Firewall	3/4	security

Layer Details

- **Data Link:** framing, CRC/checksum/parity, Hamming, ARQ, flow control, MAC, **CSMA/CD** (Ethernet), **CSMA/CA** (Wi-Fi).
- **Network:** routing (RIP, OSPF, BGP), IP, ICMP, fragmentation, congestion control, QoS. **Network = host-to-host; Transport = process-to-process.**
- **TCP:** reliable, connection-oriented, 3-way handshake (web, email, FTP). **UDP:** fast, connectionless (games, streaming, VoIP, DNS).

App	Protocol	Port
HTTP	TCP	80
HTTPS	TCP	443
FTP	TCP	20/21
SMTP	TCP	25
DNS	UDP	53
DHCP	UDP	67/68

- App-layer protocols: HTTP/HTTPS, FTP, SMTP (send mail), POP3/IMAP (receive), DNS (name→IP), DHCP (gives IP), SSH, SNMP. ! Chrome/WhatsApp are NOT application layer.

IP, MAC, Flow Control

- **IPv4:** 32-bit, 4 octets (0-255), 4.3 billion. **IPv6:** 128-bit, hex. **Private:** 10.x, 172.16-31.x, 192.168.x.

Class	1st octet	Net/Host
A	1-126	N.H.H.H
B	128-191	N.N.H.H
C	192-223	N.N.N.H

- **MAC:** 48-bit (6 bytes), first 3 bytes OUI (company), last 3 device. IP can change, MAC cannot.
- **Flow control:** Stop-and-Wait (1 frame, wait ACK, slow) • **Sliding Window** (many frames, TCP) • Credit-based.
- **Mobile:** 1G analog voice; 2G digital SMS (GSM); 3G internet; 4G LTE HD video; 5G 1-20 Gbps, IoT.
- **Internet history:** ARPANET 1969 → TCP/IP 1 Jan 1983 → DNS 1984 → WWW 1991 (Tim Berners-Lee).

4. Software Engineering (SE)

Basics

- **SE** = systematic, disciplined, measurable approach to develop, operate, maintain software (term: NATO 1968).
Software crisis (60s-70s): late, over budget, poor quality.
- Software is developed (not manufactured), does not wear out, intangible, complex, easy to modify.
- **Metrics**: Product (LOC, FP, defect density) • Process (time, cost, error detection rate).

SDLC: Requirement → Design → Coding → Testing → Deployment → Maintenance

SDLC Models

Model	Idea	Use when
Waterfall	linear sequential; next phase after previous	clear requirements, small, stable (payroll, library)
Prototype	build working model first, user feedback ("show first")	unclear requirements, UI heavy
Spiral	risk-driven iterative (Barry Boehm); Planning, Risk Analysis, Engineering, Evaluation	big, risky projects (banking, defense)

- **Waterfall**: + simple, documented; - changes hard, late testing. **Prototype**: Throwaway vs Evolutionary; + clear requirements; - time/cost. **Spiral**: + risk handling; - costly, needs experts.

Feature	Waterfall	Prototype	Spiral
Risk	Low	Medium	High
Flexibility	Low	High	High
Cost	Low	Medium	High

- **Maintenance types**: **Corrective** (fix bugs) • **Adaptive** (new OS/hardware) • **Perfective** (new features/performance).

Estimation

- **LOC**: simple but language dependent. **FP (Function Point)**: inputs, outputs, interactions, files; language independent.
Delphi: experts estimate independently, then discuss.

Basic COCOMO: Effort $E = a (KLOC)^b$ | Time $T = c (E)^d$

Type	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

EXAM TIP Example: 10 KLOC organic → $E = 2.4 \times 10^{1.05} \approx 27$ person-months, $T = 2.5 \times E^{0.38} \approx 9$ months. Intermediate = cost drivers; Detailed = per phase.

- **Halstead**: operators/operands → length, volume, difficulty (code complexity). **Putnam**: Rayleigh curve staffing (few → many → few).
- **SRS**: document of functional + non-functional requirements. Good SRS: Correct, Complete, Consistent, Unambiguous, Verifiable, Modifiable. Sections: Introduction, Overall description, Specific requirements, External interfaces.

Design, Coding, Testing

- **HLD** = architecture, modules. **LLD** = internal logic, algorithms. **Top-down** (system → modules), **Bottom-up** (modules → system), Hybrid.
- **DFD symbols**: Circle = process, Arrow = data flow, Rectangle = external entity, Parallel lines = data store. Levels: Context (0), Level 1, Level 2. Structure chart = module hierarchy.
- **OOD**: Class (blueprint), Object, Encapsulation, Inheritance, Abstraction, Polymorphism. Function-oriented = top-down (C); Object-oriented = bottom-up, reusable (Java, C++).
- **Testing**: Unit (modules, by developer) • **Black box** (input-output, no code knowledge, tester) • **White box** (logic, loops, developer) • System testing (whole system). **Debugging** = fixing found errors. Static analysis = without running; Dynamic = by running.

Quality & Maintenance

- **Reliability** = probability of failure-free operation. **Musa model**: failure rate falls with testing.
- **SQA**: reviews, audits, testing, standards. **ISO 9000**: international quality management standard (certification, no levels). **SEI CMM**: process maturity — 1 Initial, 2 Repeatable, 3 Defined, 4 Managed, 5 Optimizing.
- **Maintenance models**: Quick-fix (fast, poor quality) vs Iterative enhancement (step-by-step, better). **Reverse engineering** = recover design/requirements from existing software. Maintenance cost often > development cost (e.g. 60-70%).

5. Object Oriented Programming (C++)

Fundamentals

- **Program** = passive instructions on disk; **Process** = program in execution (active, in RAM). All processes are programs, not vice versa.
- **Machine** (binary, no translator) → **Assembly** (mnemonics, assembler, 1949) → **High level** (compiler/interpreter) ; **C** = middle level.

Basis	Compiler	Interpreter
Works	whole program at once	line by line
Speed	fast	slow
Object file	saved	not saved
Debugging	difficult	easy
Languages	C, C++, C#, Swift	Python, JS, PHP, Ruby

- Java = compiled + interpreted. **C**: Dennis Ritchie, 1972, Bell Labs, procedural. **C++**: Bjarne Stroustrup, 1979/1985, "C with Classes", .cpp, case-sensitive.

C++ Basics

- **Structure**: comments, #include <iostream>, using namespace std, global declaration, function declaration, **main()** (required), user functions.

first.cpp → Preprocessor → Compiler → .obj → Linker → .exe
→ Loader → Execution

- Turbo C++: Alt+F9 compile, Ctrl+F9 run, Alt+F5 output. C++ has 95 keywords.
- **Scope resolution ::** → global variable (::x), define member function outside class, namespace (std::cout), static member.
- **cout** << (insertion) • **cin** >> (extraction, stops at space) • cerr (unbuffered error) • clog (buffered log) • cin.get() char incl. space • cin.getline(arr,n) • getline(cin,str) best. **endl** = newline + flush (slow); **\n** fast.

Class, Object, Constructor

- **OOP** = data + functions as objects. 4 pillars: **Encapsulation, Abstraction, Inheritance, Polymorphism**. Pros: reusability, security, maintenance. Cons: more memory, slower.
- **Class** = blueprint (no memory until object). **Object** = instance (own copy of data).

Specifier	Access
public	anywhere
private	same class only (default in class)
protected	class + derived class

- **Constructor**: same name as class, no return type, auto-called, initializes. Types: **Default, Parameterized, Copy** (Student(Student &s)), overloading. Cannot be virtual/static.

- **Destructor** ~Name(): auto at scope end, no params, only one, frees memory.
- **this** = pointer to current object (resolves name clash). **Static member**: one copy shared by all objects, accessed ClassName::, static function has no this.

Inheritance & Polymorphism

- **Inheritance**: derived class gets base class properties (class D : public B). Private base members not accessible.

Type	Meaning
Single	1 base → 1 derived
Multiple	many bases → 1 derived
Multilevel	chain A→B→C
Hierarchical	1 base → many derived
Hybrid	mix (diamond)

- **Diamond problem** (two copies of base) → solve with **virtual inheritance**.
- **Compile-time (static, early) polymorphism**: **function overloading** (same name, different params; return type alone not enough), **operator overloading**. **Run-time (dynamic, late)**: **virtual function + overriding** (same signature, needs inheritance).

EXAM TIP Overriding = runtime, inheritance, same signature, virtual. Overloading = compile time, different params.

Abstraction, Exceptions, Files

- **Abstraction** = hide implementation, show essentials. **Abstract class** has ≥1 **pure virtual function** (virtual void f() = 0), no objects. **Encapsulation** = wrap data+functions (class + private). **Data hiding** = private members + getters/setters.
- **Exception handling**: try (risky code) → throw (raise) → catch (handle). Multiple catch: only one runs, checked top to bottom; catch(...) = any.
- **File handling** <fstream>: ofstream (write), ifstream (read), fstream (both). Modes: ios::in, out, app (append), trunc, binary. open(), close().

Templates & STL

- **Template** = generic code (template <class T>); function template, class template; checked at compile time.
- **STL** = Containers + Algorithms + Iterators. **Sequence**: vector, list, deque, array, forward_list. **Associative** (sorted): set, multiset, map, multimap. **Unordered** (hash, C++11): unordered_set/map.
- **Algorithms** (<algorithm>): sort, find, count, reverse. **Iterators**: begin(), end(); Input, Output, Forward, Bidirectional (list, set, map), Random access (vector, deque).

6. Data Structure (DS)

Basics & Arrays

- **Linear:** Array, Stack, Queue, Linked list (sequential, 1 level).
Non-linear: Tree, Graph (hierarchical).
- **Array:** same type, contiguous memory, index from 0, fixed size, random access $O(1)$; insert/delete needs shifting.

$$1D: a[k] = B + W(k - L) \mid \text{Elements} = U - L + 1$$

$$2D \text{ Row-major: } A[i][j] = B + W[(U_2 - L_2 + 1)(i - L_1) + (j - L_2)]$$

$$2D \text{ Column-major: } A[i][j] = B + W[(U_1 - L_1 + 1)(j - L_2) + (i - L_1)]$$

EXAM TIP Example: $A[3][4]$, $B=1000$, $W=2$, $A[2][3] \rightarrow 1000 + 2 \times [(4 \times 2) + 3] = 1022$.

- **Sparse matrix:** mostly zeros $\rightarrow$ store Row, Column, Value arrays.

Linked List

- **Node** = data + next pointer; dynamic size, insert/delete $O(1)$, access $O(n)$, extra memory.

Type	Feature
Singly	last node $\rightarrow$ NULL
Circular	last node $\rightarrow$ first node
Header	special header node, fixed reference
Doubly	prev + next, both directions

- Array vs LL: contiguous vs non-contiguous; fixed vs dynamic; access $O(1)$ vs $O(n)$. **Polynomial** node = coefficient, exponent, link.

Stack, Recursion

- **Stack** = LIFO, one end (TOP). Ops: **PUSH** (overflow if TOP = N-1), **POP** (underflow if TOP = -1), **PEEK**, **isEmpty**, **isFull**. Array (static) or linked list (dynamic).

Notation	Operator position
Infix	between: $A + B$
Prefix (Polish)	before: $+ A B$
Postfix (RPN)	after: $A B +$

- Postfix: no brackets, easy for computers/ALU. $(A + (B * C + D)) / E \rightarrow ABC * D + E / +$.
- **Recursion:** function calls itself; needs **base case** (stop) + recursive case; uses call stack (overflow risk). Factorial(4) = 24. **Iteration** = loops (for, while, do-while; break, continue) – faster, less memory.

- **Fibonacci:** 0,1,1,2,3,5,8... **Tower of Hanoi:** min moves = $2^n - 1$.

Queue

- **Queue** = FIFO; insert at **Rear**, delete at **Front**. Enqueue overflow $R = N-1$; Dequeue underflow $F = -1$.
- **Circular queue:** $R = (R+1) \% N$, full if $(F == 0 \ \&\& \ R == N-1) \parallel F == R+1$. **Deque:** insert/delete both ends (input-restricted / output-restricted). **Priority queue:** highest priority first, same priority $\rightarrow$ FIFO.

Trees

- Root, Edge, Parent, Child, **Leaf** (no child), Degree = number of children, Level, Path length = edges.
- **Binary tree:** ≤ 2 children. Array: left = $2i$, right = $2i+1$. Linked: data, left, right.

Traversal	Order
In-order	Left, Node, Right (BST $\rightarrow$ sorted)
Pre-order	Node, Left, Right
Post-order	Left, Right, Node

- **BST:** Left < Root < Right. **AVL:** self-balancing BST, **BF** = **height(L) - height(R)** $\in \{+1, 0, -1\}$. Rotations: LL $\rightarrow$ Right, RR $\rightarrow$ Left, LR $\rightarrow$ Left-Right, RL $\rightarrow$ Right-Left.

Operation	BST avg/worst	AVL
Search/Insert/Delete	$O(\log n) / O(n)$	$O(\log n)$

- **Heap** = complete binary tree; Max heap (root largest), Min heap (root smallest). **Strictly binary:** every non-leaf has 2 children; N leaves $\rightarrow 2N - 1$ nodes.

Graph & Hashing

- **Graph G = (V, E):** directed/undirected, weighted, cyclic/acyclic, complete ($n(n-1)/2$ edges). Representation: **Adjacency matrix** ($n \times n, 1/0$) / **Adjacency list**.
- **BFS** = level by level, **Queue**. **DFS** = deep then backtrack, **Stack**.
- **Hashing:** key $\rightarrow$ hash function $\rightarrow$ index; search **$O(1)$** avg (unsorted array $O(n)$, sorted $O(\log n)$, AVL $O(\log n)$).
- Hash functions: **Division** $h = \text{key} \% \text{size}$ ($27 \% 10 = 7$) • Mid-square • Folding.
- **Collision** (2 keys same index): Linear probing (+i), Quadratic (+i²), Double hashing ($h_1 + i \cdot h_2$), **Chaining** (linked list per index).

7. Theory of Computation (ToC)

Basics

- **Alphabet Σ** = finite non-empty symbols. **String** = sequence of symbols, ϵ = empty ($|\epsilon| = 0$). **Language** = set of strings. **Grammar $G = (V, T, P, S)$** = Variables, Terminals, Productions, Start.

Finite Automata

- **DFA** $M = (Q, \Sigma, \delta, q_0, F)$, $\delta: Q \times \Sigma \rightarrow Q$ — exactly ONE transition per state & symbol. $L(M)$ = all strings accepted. Double circle = final state.

Machine	Transitions
DFA	exactly one
NFA	zero, one or many
ϵ -NFA	can move without consuming input

- NFA accepts if **at least one path** reaches final state. **NFA $\equiv$ DFA** (same power = Regular languages); convert by **subset construction**.
- **Minimization**: merge equivalent states (partition, table-filling, Myhill-Nerode). Distinguish strings with a suffix that accepts one and rejects the other.

Regular Expressions & Languages

- RE operators: Union (+), Concatenation, Kleene star (*). **Precedence**: * > concatenation > +. Laws: $R + \emptyset = R$, $R\epsilon = R$, $R\emptyset = \emptyset$, $R + R = R$, $R^* = \epsilon + RR^*$.
- **Kleene's theorem**: $RE \equiv FA$. $RE \rightarrow \epsilon$ -NFA $\rightarrow$ DFA; DFA $\rightarrow$ RE by state elimination or **Arden's theorem**: $X = AX + B \Rightarrow X = A^*B$ ($\epsilon \notin A$).
- **Non-regular**: $\{0^n1^n\}$ (FA can't count). **Pumping lemma (regular)**: $w = xyz$, $|x| \leq p$, $|y| > 0$, $xy^iz \in L \forall i \geq 0$ — used to prove NOT regular.
- Regular closed under: union, intersection, complement, difference, concatenation, star, reversal. Decidable: membership, emptiness, finiteness, equivalence, universality.

Machine	Output depends on
Moore	current state only
Mealy	state + input (faster, fewer states)

- FA uses: lexical analysis, pattern matching, text search, protocol verification. Limit: no unlimited counting, no balanced parentheses.

Context Free Grammar

- **CFG** productions $A \rightarrow \alpha$. $S \rightarrow aSb \mid \epsilon$ generates $\{a^n b^n\}$. Every regular language is CFL, not every CFL is regular.

- **Derivation**: leftmost / rightmost; **parse tree** shows structure. **Ambiguous** grammar = string with >1 parse tree ($id+id*id$); **inherently ambiguous** language = no unambiguous CFG exists. Fix by precedence/associativity.
- **Simplification**: remove ϵ -productions, unit productions, useless symbols (non-generating, unreachable).

Normal form	Production
CNF	$A \rightarrow BC$ or $A \rightarrow a$ (two variables)
GNF	$A \rightarrow a\alpha$ (starts with terminal)

- CFL closed under union, concatenation, star; **NOT** under intersection, complement, difference. Emptiness, finiteness, membership decidable; equivalence **undecidable**. **Pumping lemma (CFL)**: $z = uvwxy$, $|vwx| \leq p$, $|vx| > 0$, $uv^iwx^iy \in L$.

Pushdown Automata

- **PDA = FA + Stack**; recognizes CFL. $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, Γ = stack alphabet, Z_0 = initial stack symbol. **ID** = (state, remaining input, stack).
- Acceptance: **final state** or **empty stack** (equivalent for NPDA). **DPDA** weaker than NPDA (recognizes DCFL). **CFG $\equiv$ NPDA** (CFG $\rightarrow$ PDA and PDA $\rightarrow$ CFG possible). **Two-stack PDA $\equiv$ Turing machine**.

Turing Machine & Decidability

- **TM** $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$: infinite tape, read/write head, finite control. Can accept, reject or **loop forever**. Variants (multi-tape, multi-head, non-deterministic) have same power. **UTM** simulates any TM.
- **Church-Turing thesis**: anything algorithmically computable is computable by a TM (a thesis, not theorem).

Class	Property
Recursive (decidable)	TM always halts
Recursively enumerable	accepts members, may loop on non-members

EXAM TIP Every recursive language is RE; not every RE is recursive.

- **Undecidable**: Halting problem (Turing), TM acceptance, TM equivalence, emptiness, **PCP** (Post Correspondence Problem), **MPCP** (modified PCP, used to prove PCP undecidable).

Regular $\leftrightarrow$ FA | CFL $\leftrightarrow$ PDA | Recursive/RE $\leftrightarrow$ Turing Machine

8. Computer Organization & Architecture (COA)

Basics

- **Architecture** = WHAT computer does (instruction set, visible to programmer). **Organization** = HOW (hardware implementation, control signals).
- **5 units**: Input, Output, Memory (RAM/ROM primary, HDD secondary), **Control Unit** (controls operations), **ALU** (arithmetic + logic). **CPU** = ALU + CU + Registers.

Architecture	Features
Von Neumann	von Neumann 1945; stored program; ONE memory & bus for data+instructions; bottleneck
Harvard	separate memory & buses; faster, costly
Modified Harvard	separate caches, shared main memory (Intel, AMD)

- **Von Neumann bottleneck**: one bus → one transfer at a time → CPU waits. Types: CPU, Memory, I/O bottleneck.

Fetch → Decode → Execute → Store

Memory Hierarchy

Registers → Cache → RAM → Hard Disk (speed ↓, capacity ↑, cost/bit ↓)

Memory	Speed	Cost	Capacity
Registers	fastest	highest	bytes
Cache	very fast	high	KB-MB
RAM	medium	moderate	GB
Hard Disk	slow	cheapest	TB

Error Detection & Correction

- Errors: single-bit, multiple-bit, **burst**. Detect: **Parity**, **Checksum**, **CRC**. Correct: **Hamming code**, ECC.
- **Parity**: even parity → total 1's even. Data 1010110 (four 1's): even → parity 0 → 10101100; odd → parity 1 → 10101101. Detects single-bit only.

- **Checksum**: add blocks (1's complement addition), complement the sum. 1010 + 1100 = 10110 → wrap carry 0111 → complement = **1000**. Receiver: sum all = all 1's → OK.
- **CRC**: modulo-2 (XOR) division by generator; append remainder. Data 1101, generator 1011 → append 000 → remainder **001** → send **1101001**. Receiver remainder 0 → OK. Detects burst errors.
- **Hamming**: parity bits at positions 1, 2, 4, 8...; $2^Ar \geq m + r + 1$. Data 1011 → codeword **0110011** (P1=0, P2=1, P4=0). Error position = sum of failed parity positions (received 0110111 → 4+1 = **5**). Corrects **1-bit** errors.

Technique	Detect	Correct
Parity / Checksum / CRC	Yes	No
Hamming	Yes	Yes (1 bit)

Microprocessor

- **Microcomputer** = microprocessor (CPU) + memory (RAM temporary, ROM permanent, cache) + I/O. **System bus**: Address, Data, Control.

Year	Processor	Bits
1971	Intel 4004	4 (first)
1972-74	8008, 8080	8
1978	8086	16 (x86 base)
1982	80286	16 (protected mode)
1985	80386	32 (multitasking)
1989	80486	32 (built-in FPU)
1993	Pentium	32 (dual pipeline)
2006	Core	64 (multi-core)
2008+	Core i7	64 (hyper-threading)

- Generations: 1st 4-bit, 2nd 8-bit, 3rd 16-bit, 4th 32-bit, 5th 64-bit.

9. Artificial Intelligence (AI)

Basics & Problem Solving

- **AI** = systems doing tasks needing human intelligence. ! AI = Learning + Reasoning + Problem solving + Decision making. Goals: reasoning, learning, problem solving, knowledge representation, planning, perception, NLP.
- **Problem formulation**: Initial state, Actions, Transition model, Goal state, Path cost. Flow: Initial → Actions → Goal test → Path cost.
- **Production system** = rules (IF-THEN) + working memory + control strategy. Types: **Monotonic** (actions never invalidate later rules), **Non-monotonic**, **Partially commutative** (order can change), Specialized (domain specific).
- **Search**: Uninformed (BFS, DFS, Uniform cost) • Informed (Greedy, **A***, Hill climbing). **Problem graph**: states = nodes, actions = edges. **Matching** = compare facts with rule conditions. **Indexing** = organize rules for fast retrieval.

Heuristic $h(n)$ = estimated cost from state to goal

Search Algorithms

- **Hill climbing**: local search, always moves to better neighbour; problems: local maximum, plateau, ridge.

	DFS	BFS
Strategy	deepest branch first	level by level
Structure	Stack / recursion	Queue
Memory	lower	higher
Complete / shortest	not always	yes (equal step cost)

EXAM TIP Graph $A \rightarrow (B,C)$, $B \rightarrow (D,E)$: DFS = A B D E C; BFS = A B C D E.

- **CSP** = Variables + Domains + Constraints (map coloring, Sudoku, scheduling). Techniques: Backtracking, Forward checking, Constraint propagation, Arc consistency.
- **Performance measures**: Completeness, Optimality, Time, Space complexity.
- **Game playing**: Game tree (node = state), Minimax, Alpha-Beta pruning, evaluation function.

Knowledge & Logic

- **KR techniques**: logic, semantic networks, frames, production rules, ontologies. **Predicate logic**: $\text{Human}(x) \rightarrow \text{Mortal}(x)$; quantifiers $\forall$ (for all), $\exists$ (exists).

- **Resolution**: inference rule using complementary literals; used in automated theorem proving.
- **Semantic network** = graph of concepts; **Frame** = object with attributes (Car: color, wheels); **Script** = sequence of events. **Inference** = new knowledge from facts + rules.

Reasoning & Uncertainty

	Forward chaining	Backward chaining
Driven by	Data (facts)	Goal
Direction	Facts → Conclusion	Goal → Facts
Use	Monitoring	Diagnosis, query

- **Fuzzy logic**: truth degree 0 to 1 ($30^\circ\text{C} \rightarrow \text{Hot } 0.6$; washing machine, AC). **Certainty factor** = confidence (Flu 0.8). **Bayesian theory** = probability updated with evidence. **Bayesian network** = DAG + conditional probabilities. **Dempster-Shafer** = Belief, Plausibility, Evidence; separates evidence from uncertainty.

Planning, ML, Expert Systems

- **Planning** = action sequence to reach goal. **STRIPS** (Stanford Research Institute Problem Solver): Preconditions, Add effects, Delete effects. Explanations: **Why? Why not? How?**

ML type	Idea	Example
Supervised	labelled data	spam filter; Regression, Decision tree, KNN, SVM
Unsupervised	no labels, find groups	K-Means customer groups
Reinforcement	reward / penalty	game AI
Adaptive	adjusts to new data	recommendation

- **Expert system** = Knowledge base + **Inference engine** + Working memory + User interface + Explanation facility + Knowledge acquisition module. Roles: diagnosis, prediction, planning, monitoring, design.
- **Knowledge acquisition bottleneck** = hard to collect expert knowledge. **Meta-knowledge** = knowledge about knowledge. **Shell** = expert system without domain knowledge. Examples: **MYCIN** (medical diagnosis), **DART** (logistics planning), **XCON** (computer configuration).

10. Web-Based Application Development (WAD)

Internet & Web

- **Internet** = global network of networks using **TCP/IP**. **WWW** = web pages on the Internet (WWW ⊂ Internet). **IPv4** 32-bit (4.3 billion), **IPv6** 128-bit. **DNS**: domain name → IP. **URL** = protocol + domain + path + query (https://example.com/products?id=10).
- **Front-end** (HTML, CSS, JavaScript, jQuery) = what user sees. **Back-end** (Node.js, PHP, Python, Java, database) = server processing.

User → Browser → HTTP/HTTPS → Web server → Application → Database

Tools

- **Node.js** = JavaScript runtime outside browser (V8 engine; open source, event-driven, non-blocking, asynchronous). ! Node.js is NOT a language. Run: node app.js
- **VCS** tracks file changes/versions. **Git** = distributed VCS (local, no GitHub needed). **GitHub** = cloud platform hosting Git repos (pull request, issues, forks, code review).
- Commands: **git init**, **git status**, **git add .**, **git commit -m "msg"**, **git push**, **git pull**.
- **Dreamweaver** (Adobe) = web design/development tool/IDE, not a programming language.

HTML, CSS, JS

- **HTML** = HyperText Markup Language (markup, NOT programming). Tag = <p>; element = opening tag + content + closing tag; attribute = extra info (href).
- Structure: <!DOCTYPE html>, <html>, <head> (title/metadata), <body> (visible). Tags: h1 (largest heading), p, a (link), img, br, hr, ul/ol/li, table, form.
- **Form**: **action** = where to send; **method** = GET (data in URL) / POST (data in body).
- **CSS** = presentation (color, font, size, margin, padding, border, layout). Types: **Inline** (style="..."), **Internal** (<style>), **External** (<link rel="stylesheet">).
- **JavaScript** = high-level language for dynamic, interactive pages. **jQuery** = JavaScript library: \$(selector), .click(), .hide(), .show(), .css(), .html(), .text(), .val(), .addClass().

- **DOM** = document as object tree; getElementById(), getElementsByClassName(), querySelector(), querySelectorAll(), createElement(), remove().

XML, HTTP, Session

- **XML** = store/transport data; extensible (own tags), self-descriptive, platform independent, hierarchical, **case-sensitive**. **DTD** defines & validates XML structure (<!ELEMENT student (name, age)>).
- **HTTP** = application-layer request/response. Methods: **GET** (retrieve), **POST** (submit), **PUT** (update), **DELETE**. HTTP is **stateless**.

	Cookie	Session
Stored at	client browser	server
Data	small	more state
Identify by	sent with requests	Session ID
Use	preferences, theme	logged-in state

- **Client-side** (browser: JavaScript) vs **Server-side** (server: PHP, DB access).

PHP & MySQL

- **PHP** = PHP: Hypertext Preprocessor, server-side scripting. <?php \$name = "Rahul"; echo "Hello " . \$name; ?>
- **SQL** = language (SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP). **MySQL** = open-source RDBMS software. ! SQL = language, MySQL = DBMS.
- **PHP + MySQL** via **MySQLi** or **PDO**: \$conn = new mysqli("localhost","root","","school"); then \$conn->query(\$sql) for INSERT / SELECT / UPDATE / DELETE; fetch_assoc() reads rows.

Form → JS validation → HTTP POST → PHP → MySQL INSERT → PHP result → HTTP response → Browser

11. Multimedia

Basics

- **Multimedia** = 2+ media (text, graphics, image, audio, video, animation). **Linear** (no user control: movie, TV) vs **Non-linear** (interactive, user navigates: website, e-learning).
- Applications: education, entertainment, business, medicine (3D anatomy), advertising, science, e-commerce, communication (video conferencing).

Hardware, Software, Storage

- **Hardware**: PC, monitor, speakers, microphone (sound → electrical), camera, **scanner** (document → digital image), projector, **sound card**, **graphics card/GPU**, storage.
- **Software**: Photoshop/GIMP (image), Audacity (audio), Premiere Pro/DaVinci (video), Blender/Adobe Animate (animation), PowerPoint, VLC.

	CD-ROM / CD	DVD
Full form	Compact Disc	Digital Versatile Disc
Capacity	~700 MB (read-only CD-ROM)	4.7 GB+ (DVD-5 4.7, DVD-9 8.5, DVD-10 9.4, DVD-18 17 GB)
Use	music, software	movies, games, backup

Audio & Text

- Recording: Sound → Mic → **ADC** → digital → storage; playback: digital → **DAC** → speaker.
- **Sampling rate** = samples per second (CD = 44.1 kHz = 44,100/s). **Bit depth** = bits per sample (8, 16, 24; higher = more accurate). **MP3** = lossy. **FLAC** = lossless. **WAV** = uncompressed.
- **MIDI** = musical INSTRUCTIONS (note, velocity, instrument), tiny files, no actual sound. **Digital audio** = actual recorded sound. Editing: cut, trim, fade-in/out, noise reduction, mixing.
- **Text**: Static (fixed), Dynamic (changing: live score), **Hypertext** = text + links.

Graphics & Imaging

- **RGB** (additive, screens) vs **CMYK** (subtractive, printing). **Pixel** = smallest image element; 1920×1080 = 2,073,600 pixels. Resolution = detail.

	Raster	Vector
Made of	pixel grid	math (lines, curves)
Examples	photos, scans	logos, icons
Formats	JPEG, PNG, GIF, BMP, TIFF	SVG, EPS

- **JPEG** lossy (photos) • **PNG** lossless + transparency • **GIF** 256 colors + animation • **BMP** uncompressed • **TIFF** high quality (print/scan) • **SVG** scalable vector.
- **Scanner** = documents; **digital camera** = lens + sensor, captures scenes (JPEG, RAW).

Animation & Video

- **Frame, FPS** (24 FPS = 24 frames/s), **Keyframe** (important state), **Tweening** = in-between frames. Techniques: frame-by-frame, keyframe, path-based, **sprite** (2D object), **motion capture**.
- **FK** (Forward Kinematics): parent joint movement → end position. **IK** (Inverse): desired end position → joint movement. **Morphing** = one image transforms into another. Blender = 3D, Adobe Animate = 2D, Maya = 3D.

Std	Region	FPS
NTSC	USA, Japan	~29.97
PAL	India , Europe	25
SECAM	France	25

- Video resolution: 1280×720 HD, 1920×1080 Full HD, 3840×2160 4K. **Bit rate** = data per second. Codecs H.264/H.265. Formats MP4, AVI, MOV, MKV, WebM.
- **Video production**: **Pre-production** (planning, script, storyboard) → **Production** (shooting) → **Post-production** (editing, effects, export).

Project & Future

Idea → Planning → Design → Development → Authoring → Testing → Delivery → Maintenance

- **Authoring** = combining media elements into an interactive application. Team: project manager, writer, designer, animator, audio/video specialist, programmer, UI/UX, tester.
- **Interactive TV** (on-demand, menus, voting) • **Digital broadcasting** (better quality, more channels) • **Digital radio** • **Multimedia conferencing** (audio + video + screen share; camera + mic + speakers + network + software).

12. Internet of Things (IoT)

Fundamentals

- **IoT** = network of physical things with sensors, software, connectivity exchanging data. ! IoT = Things + Sensors + Connectivity + Data + Intelligence.
- **Functional requirements:** Sensing, Connectivity, Data processing, Identification (IP/MAC/RFID), Actuation, Security.

Layer	Function
1 Perception	sensors collect data (RFID, camera)
2 Network	data transfer (Wi-Fi, Zigbee, 5G)
3 Processing / Middleware	store, process, analyze
4 Application	smart home, city, agriculture
5 Business	decisions, policies

- **Four pillars:** People, Process, Data, Things. **DNA of IoT:** sensing, communication, identification, data processing, intelligence, actuation, security. **Web 3.0** = intelligent, semantic, decentralized web. **Toolkit approach** = users build IoT apps without programming.

Middleware & Protocols

- **Middleware** = software layer between devices and applications (device/data mgmt, security, discovery). **Communication middleware:** MQTT, CoAP, AMQP, DDS.

Protocol	Meaning
M2M	machine-to-machine, no human
WSN	wireless sensor network (Zigbee, 6LoWPAN)
SCADA	Supervisory Control & Data Acquisition (industrial)
RFID	radio waves; tag + reader + backend
IEEE 802.15.4	low-rate WPAN standard (Zigbee base)
BACnet	building automation (HVAC, lighting)
Modbus	industrial protocol: RTU, ASCII, TCP
KNX	open building automation standard

- **Zigbee:** low power, low cost, short range, **mesh**. Devices: **Coordinator** (creates network), **Router** (forwards), **End device**. **APS** = application support sublayer (binding, address mapping, groups). Has encryption & authentication.

Security, WoT, Cloud

- **Security:** **CIA** = Confidentiality, Integrity, Availability. Threats: unauthorized access, malware, eavesdropping, data theft, device hijacking, DDoS.
- **Standardization:** needed for **interoperability** (also compatibility, security, scalability). Unified data standards = common data format.
- **WoT** = IoT devices accessed via Web technologies (HTTP, REST, Web APIs). Pillars: **URI** (identifies resource) + **HTTP** (communication). Multitier: Thing → Communication → Middleware → Application → Business. WoT portal + Business Intelligence for decisions.
- **Cloud of Things** = IoT + Cloud. Grid computing, **SOA** (reusable services), cloud middleware. Providers: **AWS, Azure, Google Cloud**. Mobile cloud computing offloads processing.

IIoT & Advanced

Industry	Key technology
1.0	steam power
2.0	electricity, mass production
3.0	electronics, automation
4.0	cyber-physical systems, IoT, AI, data

- **IIoT:** smart industrial machines — predictive maintenance, smart manufacturing, asset tracking, quality control. Challenges: cybersecurity, cost, legacy systems, interoperability.
- **Security** = protect from cyber threats; **Safety** = protect from physical hazards. **Secure-by-design; run-time monitoring; ARMET** = runtime monitoring/dependability. **Dependability** = reliability, availability, safety, security, maintainability.
- **Clustering** = group sensors, **cluster head** sends data (saves energy). **Synchronization** = common timing. **Software agent** = autonomous task-doing software. **Smart grid** = two-way, real-time, automated electricity network (smart meter). **IoT EV charging** = remote monitoring, smart charging, payment, load balancing.

13. E-Commerce

Basics & Models

- **E-Commerce** = buying/selling goods, services, information, money over electronic networks. Flow: Customer → Internet → Merchant → Online payment → Delivery.
- **Characteristics**: electronic, global reach, 24×7, paperless, fast, lower cost, convenient, personalization.

Model	Meaning	Example
B2B	business → business	manufacturer → wholesaler
B2C	business → consumer	Amazon
C2C	consumer → consumer	OLX
C2B	consumer → business	freelancer, photographer
B2G	business → govt	e-tender, IT systems
G2B	govt → business	GST, business registration
G2C	govt → citizen	passport, certificates
C2G	citizen → govt	tax payment, complaints

Framework, Internet, I-Way

- **Framework**: Network infrastructure, Information distribution (WWW), Messaging (email, EDI), Security, Payment, Applications.
- **I-Way** (Information Superhighway) = high-speed digital infrastructure: Consumer access equipment • Local access networks • Global information distribution network (backbone, fiber, satellite).
- **Internet** = network of networks; **WWW**; browser; web server; **URL**; **IP address**; **Domain name**; **ISP**. **TCP** = reliable delivery; **IP** = addressing + routing. **NSFNET** = US National Science Foundation network that expanded the Internet. **ISOC** promotes open Internet; **IETF** = standards; **ICANN** = names & numbers.

Security

- **CIA**: Confidentiality, Integrity, Availability. Threats: Malware (virus, worm, trojan, ransomware), Phishing, Password attack, DoS/DDoS, Man-in-the-Middle.
- **Firewall** = monitors/controls traffic by rules (Internet → Firewall → Internal network). Types: Packet filtering, Stateful, Proxy, Next-generation.
- **Encryption**: Plaintext → Ciphertext (Decryption reverses). **Symmetric** (same key, fast, AES; key distribution problem) vs **Asymmetric** (public + private key, RSA, slower).

- **Digital signature** = hash signed with private key, verified with public key; checks authenticity + integrity. ! ≠ scanned signature.
- **HTTPS = HTTP + TLS** (encryption, authentication, integrity). **SSL** = older protocol; **TLS** = modern successor.

Consumer E-Commerce & Payment

Mercantile model: Search → Select → Purchase decision → Order → Payment → Delivery → After-sales service

- Consumer wants quality, price, security, fast delivery, refund; merchant wants more customers/sales, secure payment, low cost.
- **E-payment**: credit/debit card, internet banking, UPI, wallet, e-cash, smart card. **Digital token** = value/authorization representation (card number → token). **E-cash** = digital money; concern: **double spending**.
- **Smart card**: chip card — Memory card (stores) vs Microprocessor card (processes): EMV, SIM.
- **Credit card**: Cardholder, Merchant, **Issuer** (bank, authorizes), **Acquirer**, Card network. Flow: Customer → Merchant → Acquirer/Network → Issuer → settlement.
- **Threats**: payment info theft, phishing, malware, MITM, **replay attack**, identity theft. **Measures**: encryption, authentication, MFA, TLS, tokenization, fraud detection.

EDI & Digital Information

- **EDI** = computer-to-computer exchange of standardized business documents (purchase order, invoice, shipping notice). Benefit: less manual entry, automation. Used in manufacturing, retail, banking, logistics, healthcare, government.
- EDI issues: **Security** (interception, non-repudiation), **Legal** (validity of e-records, contracts, digital signatures), **Privacy** (access control).

	EDI	Traditional
Documents	electronic	paper
Process	computer-to-computer	human
Speed	faster	slower

- **Corporate digital library** = centralized electronic repository. **Digital documents**: PDF, DOCX, XML, e-invoice. **Data warehouse** = integrated data for reporting/decisions: Operational DBs → ETL → Warehouse → Reports → Decisions.