Exit Exam - ChatGPT

▼ Computer Programming

Data types

- **Primitive types**: integers, floating-point numbers, characters, and boolean values
- **Composite types**: arrays, structures, and classes
- Types
  - Integer: whole numbers with no decimal point
  - Float/Double: numbers with decimal points
  - Boolean: true/false values
  - Character: a single letter, digit or symbol
  - String: a collection of characters
  - Array: a collection of data items of the same type, accessed by an index
  - **Structure**: a collection of data items of different types
  - **Pointer**: a variable that stores the memory address of another variable
  - **Enumeration**: a user-defined data type consisting of a set of named constants

Control structures

- Sequence: statements executed one after another
- Selection: statements executed conditionally based on a boolean expression
- Iteration: statements executed repeatedly based on a boolean expression

Functions

- A group of statements that perform a specific task

Memory management

- Stack: used for storing variables, function calls, and return addresses

- Heap: used for dynamic memory allocation
- Pointers: variables that store the memory address of another variable

Input/output

- Input: reading data from user input or files
- Output: printing data to the console or writing data to files

Debugging

- Finding and fixing errors in code Techniques include using print statements, debugging tools, and unit tests

Types of programming languages

1. **Procedural** programming languages: These are languages that follow a specific set of procedures or steps to complete a task. Examples include C, FORTRAN, and COBOL.
2. **Object-oriented** programming languages: These languages are based on the concept of objects, which contain both data and methods (or functions) that operate on that data. Examples include Java, C++, and Python.
3. **Functional** programming languages: These languages are based on mathematical functions and treat computation as the evaluation of expressions. Examples include Haskell, Lisp, and Scheme.
4. **Scripting** programming languages: These are interpreted languages that are used for automating tasks and for rapid prototyping. Examples include JavaScript, Python, and Ruby.
5. **Markup** languages: These languages are used to define the structure and presentation of data, such as web pages. Examples include HTML, XML, and LaTeX.
6. **Query** languages: These languages are used to extract information from databases. Examples include SQL, XQuery, and SPARQL.
7. **Domain-specific** languages: These are languages that are designed for a specific task or domain, such as music notation, financial modeling, or scientific simulations. Examples include MATLAB, R, and VHDL.
8. **Imperative** programming: focuses on describing how a program should execute

Integrated development environments (IDEs)

- Software applications that provide a comprehensive environment for coding, debugging, and testing
- Examples include Visual Studio, Eclipse, and PyCharm

Version control

- A system for managing changes to code over time
- Allows developers to work collaboratively and keep track of changes and revisions
- Examples include Git and SVN.

▼ Fundamental of Data structure and analysis

- Data structures are used to organize and store data in a way that makes it easy to access and manipulate.
- Data structures can be divided into two main categories: linear and non-linear.
    - **Linear data structures**
        - Arrays, linked lists, stacks, and queues.
        - Arrays are a collection of elements of the same type, stored in **contiguous memory locations.**
        - Linked lists, on the other hand, are made up of nodes that contain data and a pointer to the next node.
        - Stacks and queues are abstract data types that can be implemented using arrays or linked lists.
        - Stacks use the "last in, first out" (LIFO) principle, while queues use the "first in, first out" (FIFO) principle.
    - **Non-linear data structures**
        - Include trees and graphs.
        - Trees are hierarchical structures made up of nodes connected by edges.

        - Graphs are used to represent complex relationships between objects. They consist of vertices (also known as nodes) and edges, which connect the vertices.
- Data structure analysis involves the study of the performance of algorithms that operate on data structures.
    - **Time complexity** refers to the amount of time it takes for an algorithm to run, as a function of the size of the input data.
    - **Space complexity** refers to the amount of memory required by an algorithm, also as a function of the size of the input data.

> 💡 Big O represents the worst-case scenario, Big Theta represents both the upper and lower bounds, and Big Omega represents the best-case scenario.

- The "big O" notation is commonly used to express time and space complexity. It provides an upper bound on the growth rate of a function, and helps to compare the performance of different algorithms.
- **Big O notation**: It is used to describe the upper bound of the growth rate of a function. In other words, it represents the worst-case scenario for the function's growth rate. For example, if a function is said to be O(n), it means that the function's growth rate is proportional to n or less than n.
- **Big Theta notation**: It represents both the upper and lower bounds of a function's growth rate. It means that the function's growth rate is bounded both above and below by the same function. For example, if a function is said to be Θ(n), it means that the function's growth rate is proportional to n.
- **Big Omega notation**: It is used to describe the lower bound of the growth rate of a function. It represents the best-case scenario for the function's growth rate. For example, if a function is said to be Ω(n), it means that the function's growth rate is proportional to n or greater than n.

▼ Object Oriented Programming

- Object-Oriented Programming is a programming paradigm that is based on the concept of objects.
- An **object** is an instance of a class, which is a blueprint for creating objects.

- A **class** defines the properties and methods that an object will have.
- **Properties** are data members, also known as fields, and **methods** are functions that operate on the data members.
- OOP is used to create modular and reusable code. It allows for the creation of complex systems that are made up of smaller, more manageable parts.
- It is commonly used in software development to create applications with graphical user interfaces, web applications, and mobile applications.
- Some popular programming languages that support OOP include Java, Python, C++, and Ruby.

Main concepts in OOP

- **Encapsulation** refers to the practice of hiding the internal details of an object and exposing only the necessary information to the outside world. It allows the object to be accessed through a well-defined interface, while the implementation details remain hidden.
- **Inheritance** allows a class to inherit properties and methods from another class. This enables the creation of new classes that are similar to existing classes, but with additional or modified functionality.
- **Polymorphism** allows objects of different classes to be treated as if they are of the same type. This enables the creation of more flexible and reusable code.
- **Abstraction** refers to the process of focusing on the essential features of an object, while ignoring the details that are not relevant to its use. It allows the object to be viewed at a high level of abstraction, without getting bogged down in the implementation details.

▼ Java example

- In this code, we have an `Employee` class that demonstrates encapsulation, where the `id` and `name` fields are private and can only be accessed through the public `getId()` and `getName()` methods.
- The `Manager` class extends the `Employee` class, demonstrating inheritance, where it inherits the `id` and `name` fields and methods from `Employee`, but also has a `department` field and method specific to a `Manager`.

- The `Shape` interface demonstrates abstraction, where it defines two methods,

```
// Encapsulation example
public class Employee {
    private int id;
    private String name;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

// Inheritance example
public class Manager extends Employee {
    private String department;

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }
}

// Abstraction and Polymorphism example
public interface Shape {
    double getArea();
    double getPerimeter();
}

public class Rectangle implements Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    public double getLength() {
        return length;
    }
```

```
    public void setLength(double length) {
        this.length = length;
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getArea() {
        return length * width;
    }

    public double getPerimeter() {
        return 2 * (length + width);
    }
}

public class Circle implements Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double getRadius() {
        return radius;
    }

    public void setRadius(double radius) {
        this.radius = radius;
    }

    public double getArea() {
        return Math.PI * radius * radius;
    }

    public double getPerimeter() {
        return 2 * Math.PI * radius;
    }
}

public class Main {
    public static void main(String[] args) {
        // Encapsulation example
        Employee emp = new Employee();
        emp.setId(100);
        emp.setName("John");

        System.out.println("Employee ID: " + emp.getId());
        System.out.println("Employee Name: " + emp.getName());

        // Inheritance example
        Manager mgr = new Manager();
```

```
        mgr.setId(101);
        mgr.setName("Mary");
        mgr.setDepartment("HR");

        System.out.println("Manager ID: " + mgr.getId());
        System.out.println("Manager Name: " + mgr.getName());
        System.out.println("Manager Department: " + mgr.getDepartment());

        // Abstraction and Polymorphism example
        Shape rectangle = new Rectangle(5, 10);
        System.out.println("Rectangle Area: " + rectangle.getArea());
        System.out.println("Rectangle Perimeter: " + rectangle.getPerimeter());

        Shape circle = new Circle(5);
        System.out.println("Circle Area: " + circle.getArea());
        System.out.println("Circle Perimeter: " + circle.getPerimeter());
    }
}
```

▼ Web Design and Programming

- Web design and programming involves the creation and maintenance of websites and web applications.
- It involves designing the layout, choosing the colors and fonts, and creating graphics and other visual elements. The goal of web design is to create a user-friendly and visually appealing website that effectively communicates the desired message to the target audience.
- **HTML**: HTML (Hypertext Markup Language) is a markup language used to create web pages. HTML defines the structure and content of a web page, including headings, paragraphs, lists, and links.
- **CSS**: CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a web page. CSS is used to control the layout, fonts, colors, and other visual aspects of a web page.
- **JavaScript**: JavaScript is a programming language used to create dynamic and interactive web pages. JavaScript can be used to manipulate HTML and CSS, respond to user input, and communicate with servers.
- **Server-side programming**: Server-side programming is used to generate dynamic content on the web server before it is sent to the client's web browser. Common server-side programming languages include PHP, Python, Ruby, and Java.

- **Client-side programming**: Client-side programming is used to create interactive features on the client side, within the user's web browser. JavaScript is the primary language used for client-side programming.
- **Web development frameworks**: Web development frameworks provide a set of tools and libraries to streamline the development process. Popular web development frameworks include Ruby on Rails, Django, and Laravel.
- **Responsive design**: Responsive design uses **CSS media queries** and other techniques to adjust the layout and content of a web page based on the size of the user's screen.
- **User experience (UX) design**: User experience design is a design approach that focuses on creating websites and web applications that are easy to use and provide a positive experience for the user. UX design involves understanding the user's needs, goals, and behavior, and designing the user interface and user interactions to meet those needs
    1. **User experience design (UX)**: This involves designing the website in a way that makes it easy and intuitive for users to navigate and find the information they need.
    2. **User interface design (UI)**: This involves designing the visual elements of the website, such as the buttons, menus, and other interactive elements.
- **Accessibility**: This involves designing the website in a way that makes it accessible to users with disabilities.

▼ Mobile Application Development [incomplete]

▼ Common terminologies

▼ Pushing notification

▼ Android manifest

The Android architecture can be divided into four main layers:

1. Linux kernel: The bottom layer of the Android architecture is the Linux kernel, which provides basic system functionality such as process management, memory management, and device drivers. Android uses a modified version of the Linux kernel, which includes additional features such as power management and security enhancements.

2. Libraries and runtime: The second layer of the Android architecture includes libraries and runtime components that are required for application development. This includes the Android Runtime (ART), which is responsible for running and managing application code, as well as libraries for graphics rendering, database access, and networking.
3. Application framework: The third layer of the Android architecture is the application framework, which provides a set of services and APIs that developers can use to build applications. This includes components such as activities, content providers, broadcast receivers, and services, which allow developers to create interactive and responsive applications.
4. Applications: The top layer of the Android architecture is the applications layer, which includes all the user-facing applications that are installed on the device. These applications are written using the Android SDK, which provides a set of tools and APIs for developing Android applications.

▼ Fundamentals of Database

▼ Common terminologies

- Database: A collection of data that is organized in a particular way.
- DBMS (Database Management System): A software system that manages the creation, storage, retrieval, and modification of data in a database.
- RDBMS (Relational Database Management System): A DBMS that is based on the relational data model, which organizes data in tables or relations.
- SQL (Structured Query Language): A programming language that is used to manage data in an RDBMS, including creating tables, modifying data, and querying data.
- Table: A collection of data in a relational database that is organized into rows and columns.
- Row: A single record or entry in a table that represents a unique instance of an entity.
- Column: A single field or attribute in a table that represents a specific data type or value.

- Primary Key: A unique identifier for each row in a table, which ensures that each row is uniquely identifiable.
- **Foreign Key**: A key that is used to create a relationship between two tables in a relational database, by linking the primary key of one table to a column in another table.
- Index: A database structure that allows for fast searching and retrieval of data based on specific criteria.
- Query: A request for data from a database, which can be used to extract, modify, or delete data from a database.
- Backup: A copy of a database that is stored separately from the original database, in case of data loss or corruption.
- Transaction: A series of database operations that are treated as a single unit of work, which ensures that all operations are either completed or rolled back if an error occurs.
- ACID (Atomicity, Consistency, Isolation, Durability): A set of properties that ensure that database transactions are completed reliably, including atomicity (all or nothing), consistency (ensuring that the data is correct), isolation (transactions do not interfere with each other), and durability (data is permanently stored).
- Data Warehouse: A large, centralized database that is used to store and analyze large amounts of data from multiple sources, to support business intelligence and decision-making.
- Data Mining: The process of analyzing large datasets to extract patterns and insights, using machine learning and statistical techniques.
- **Database Schema**: A database schema is a blueprint that defines the structure of a database. It specifies the tables, columns, and relationships between them, as well as constraints, indexes, and other rules that govern the data stored in the database.
- Database Model: A database model is a conceptual representation of a database that defines how the data is organized and how the various components of the database relate to each other. The three most common types of database models are the relational model, the hierarchical model, and the network model.

- **Migration**: Database migration is the process of moving a database from one system or environment to another. This process may involve transferring data, schema, and other objects from one database to another.
- Lock: A database lock is a mechanism used by database management systems to control concurrent access to shared data. It is used to prevent multiple transactions from accessing the same data simultaneously, which can result in data inconsistencies and corruption.

▼ Types of databases

1. **Relational Databases**:
    - Use a tabular structure with rows and columns to store data
    - Use SQL (Structured Query Language) to access and manipulate data
      Examples: MySQL, Oracle Database, Microsoft SQL Server
2. Object-Oriented Databases:
    - Store data as objects, which include data and behavior
    - Use object-oriented programming languages to access and manipulate data
      Examples: MongoDB, ObjectDB, db4o
3. Hierarchical Databases:
    - Organize data in a tree-like structure, with each record having one parent record and many children
    - Often used for mainframe systems
      Examples: IBM Information Management System (IMS), Windows Registry
4. Network Databases:
    - Similar to hierarchical databases, but each record can have multiple parent and child records
    - Often used for complex systems such as telecommunications
      Examples: Integrated Data Store (IDS), IDMS/DB
5. Graph Databases:

- Use graph structures to store and represent data, with nodes representing entities and edges representing relationships between them
- Useful for complex and interconnected data, such as social networks Examples: Neo4j, OrientDB, ArangoDB

6. Document Databases:
   - Store data as documents, usually in JSON or XML format
   - Documents can be nested and have different structures Examples: CouchDB, MongoDB, RavenDB
7. Time-Series Databases:
   - Designed for handling time-series data, such as sensor data or financial data
   - Optimized for high write rates and fast queries over time ranges Examples: InfluxDB, TimescaleDB, OpenTSDB
8. Cloud Databases:
   - Databases hosted on cloud platforms, offering scalability and high availability
   - Can be relational, NoSQL, or specialized for specific use cases Examples: Amazon RDS, Azure SQL Database, Google Cloud Firestore.

▼ Database Normalization

> 💡 Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity.

1. First normal form (1NF): A table is in first normal form if all of its columns contain atomic (indivisible) values. This means that each cell in the table contains only one value, and not a list of values.
2. Second normal form (2NF): A table is in second normal form if it is in 1NF and all non-key attributes are fully dependent on the primary key. This means that each non-key attribute is dependent only on the primary key, and not on any other non-key attributes.

3. Third normal form (3NF): A table is in third normal form if it is in 2NF and all non-key attributes are dependent only on the primary key. This means that each non-key attribute is dependent only on the primary key, and not on any other non-key attributes.
4. Fourth normal form (4NF): A table is in fourth normal form if it is in 3NF and has no multi-valued dependencies. This means that each attribute in the table is independent of any other attributes, and there are no groups of attributes that are dependent on each other.
5. Fifth normal form (5NF): A table is in fifth normal form if it is in 4NF and has no join dependencies. This means that there are no two or more sets of attributes that can be combined to form a single attribute in the table.

▼ Database administration

- Database management systems: Database management systems (DBMS) are software applications that are used to manage and manipulate the data in the databases.
- A database system consists of a database server, database management system (DBMS), and storage subsystems.
- Different types of database architecture include client-server, peer-to-peer, and distributed database systems.
- Database installation and configuration involve setting up the DBMS and any additional tools or utilities required.
- **Backup and recovery strategies** are critical for ensuring that data can be recovered in the event of a system failure, data corruption, or other types of disasters.
- **Monitoring** tools and techniques should be used to track database performance and resource utilization in real-time.
- **Replication** is the process of copying data from one database to another, and there are different replication strategies available.
- Scaling and high availability strategies are important considerations in database administration, and can include techniques such as vertical scaling, horizontal scaling, failover, and clustering.

▼ Scaling, failover, and clustering

Vertical Scaling

- Also known as scaling up, it involves increasing the resources of a single server, such as CPU, RAM, and storage.
- This is achieved by upgrading the hardware of the server, such as adding more memory or swapping out the CPU for a faster one.
- Vertical scaling is easier to implement but has limits in terms of the maximum capacity of the server.

Horizontal Scaling

- Also known as scaling out, it involves adding more servers to a system to handle increased load.
- This can be achieved by adding more servers to a server cluster or by using a load balancer to distribute traffic across multiple servers.
- Horizontal scaling is more complex to implement, but it offers greater scalability and better fault tolerance.

Failover

- Failover is a mechanism used to provide high availability in case of server failure.
- It involves transferring the workload from a failed server to a backup server automatically.
- The backup server is usually a replica of the primary server, so it can take over quickly and seamlessly.

Clustering

- Clustering is a technique used to improve performance, scalability, and availability by grouping multiple servers together.
- In a cluster, each server works together to share the workload and provide redundancy.
- Clustering can be implemented in different ways, such as shared storage or distributed processing.

▼ Performance tuning

> 💡 Involves optimizing database systems for maximum performance, which can include techniques such as indexing, partitioning, and caching.

- Optimizing queries:
  - Use indexes to speed up query execution time
  - Use query hints to influence the execution plan of a query
  - Rewrite complex queries to make them more efficient
- Improving hardware performance:
  - Increase memory/RAM on the server to reduce disk I/O
  - Use solid-state drives (SSDs) instead of traditional hard drives
  - Increase CPU processing power
  - Add more servers to distribute workload
- Reducing network latency:
  - Minimize the distance between servers and clients
  - Use content delivery networks (CDNs)
  - Reduce the size of data being transferred
  - Use compression techniques to reduce data size
- **Indexing**:
  - Indexes are used to improve the performance of database queries by allowing the DBMS to quickly locate the data that is needed.
  - Indexes also improve query performance by reducing the number of rows that need to be scanned
  - Indexes can be created on one or more columns in a table
  - The type of index used depends on the database system being used and the types of queries being executed
- **Partitioning**:
  - Partitioning involves dividing large tables into smaller, more manageable pieces

    - This can improve query performance by reducing the amount of data that needs to be scanned
    - Partitioning can be done based on range, hash, or list
- **Caching**:
    - Caching involves storing frequently accessed data in memory to reduce the number of disk I/O operations
    - This can significantly improve performance for read-intensive applications
    - There are various caching techniques, such as in-memory caching, distributed caching, and content caching

▼ ACID Model

- **Atomicity**: Transactions are atomic, meaning they either complete in full or not at all. If any part of the transaction fails, the entire transaction is rolled back.
    - Ensures data consistency: Either all operations of the transaction are executed, or none of them are.
    - In order to achieve atomicity, database management systems typically use a technique called logging, where all changes made by a transaction are recorded in a log file.
- **Consistency**: Transactions maintain data consistency, meaning the database will be in a valid state before and after each transaction.
- **Isolation**: Transactions are isolated from each other, meaning they don't interfere with one another. Each transaction operates as though it's the only one executing, even if multiple transactions are happening at the same time.
- **Durability**: Once a transaction is committed, it is durable and will not be lost. The changes made to the database are permanent and will survive any subsequent failures.

▼ Fundamentals of Networking

▼ Basics

1. **Protocol**: A set of rules and standards that govern the communication between devices on a network.
    - The Internet Protocol **(IP)** is the most common protocol used for communicating between devices on the internet.
    - Transmission Control Protocol **(TCP)** for reliable communication
    - User Datagram Protocol **(UDP)** for fast, unreliable communication.
2. **IP Address:** A unique identifier assigned to every device on a network to allow it to communicate with other devices.
3. **MAC Address**: A unique identifier assigned to the network interface card (NIC) of a device.
4. **Gateway**: A device that connects different networks and translates data from one network to another.
5. **Bandwidth**: The maximum amount of data that can be transmitted over a network in a given time.
6. **Latency**: The amount of time it takes for data to travel from one point to another on a network.
7. **DNS**: Domain Name System is a system that translates domain names into IP addresses, making it easier for users to access websites.
8. **DHCP**: Dynamic Host Configuration Protocol is a protocol that automatically assigns IP addresses to devices on a network.
9. **VPN**: Virtual Private Network is a secure way to access a private network over a public network such as the Internet.
10. **NAT**: Network Address Translation is a technique used to map IP addresses between different networks.
11. **SSL/TLS:** Secure Sockets Layer/Transport Layer Security is a protocol used to provide secure communication over the Internet.
12. **Port**: A number used to identify a specific process or application running on a device that is communicating over a network.

▼ Network architectures

- In a **client-server** architecture, a centralized server is used to provide resources to clients. Also known as two-tier architecture.

    - Examples include web servers and email servers.
- In a **peer-to-peer** architecture, resources are shared between devices without the need for a central server. Also known as a decentralized architecture
    - Examples include file-sharing networks and Bitcoin.
- Cloud computing involves using remote servers to store, manage, and process data.
    - Examples: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) are common cloud computing architectures.
- Service-Oriented Architecture:
    - Consists of services that are provided and consumed by different components of an application or system.
    - Each service is self-contained and can be reused across different applications and systems.
    - Examples include RESTful APIs and SOAP web services.

▼ Types of networks

- **LANs** are used to connect devices in a small area, such as a home or office.
- **WANs** are used to connect devices over long distances, and can include technologies such as the internet.
- **MANs** are used to connect devices in a metropolitan area
- **SANs** are used to provide high-speed access to storage devices.

▼ Network Topologies

- Network topologies refer to the physical layout of devices in a network, such as bus, star, ring, and mesh topologies.
- In a bus topology, all devices are connected to a central cable.
- In a star topology, all devices are connected to a central hub or switch.
- In a ring topology, devices are connected in a closed loop.
- In a mesh topology, devices are connected to multiple other devices.

- Hybrid topologies are a combination of two or more different topologies.

▼ Network devices

- Network devices are hardware components that are used to connect, manage, and secure computer networks.

1. **Switches**: Switches are devices that connect network segments together and manage the flow of data between them. They operate at the data link layer of the OSI model and use MAC addresses to forward data to the correct destination.
2. **Routers**: Routers are devices that connect multiple networks together and route data packets between them. They operate at the network layer of the OSI model and use IP addresses to forward data to the correct destination.
3. **Hubs**: Hubs are devices that connect multiple devices together in a network. They operate at the physical layer of the OSI model and broadcast all data they receive to **every device** on the network.
4. **Modems**: Modems are devices that connect a computer to the internet over a telephone line or cable. They convert digital signals from the computer into analog signals that can be transmitted over the phone or cable line.
5. **Firewalls**: Firewalls are devices that **protect** a network from unauthorized access and malicious traffic. **They filter traffic** based on predefined rules and can be configured to block or allow traffic based on source, destination, protocol, and other factors.
6. **Access points:** Access points are devices that provide wireless connectivity to devices on a network. They allow devices to connect to a wireless network and communicate with other devices on the same network.
7. **Network attached storage (NAS):** NAS devices are storage devices that are connected to a network and **provide centralized storage** for files and data. They can be accessed by multiple devices on the network, making it easy to share files and data across the network.

▼ Types of network Cables

1. **Twisted Pair Cable**: This is the most common type of networking cable used in LANs (Local Area Networks). It consists of two or more pairs of wires twisted together and is used to transmit data signals. There are two types of twisted pair cables:
    - Unshielded Twisted Pair (UTP):
        - Used in most LANs and is the most common type of networking cable.
        - UTP cables have no shielding to protect against electromagnetic interference, but are less expensive and easier to install.
    - Shielded Twisted Pair (STP):
        - Has a metal shield around each pair of wires to protect against electromagnetic interference.
        - STP cables are more expensive and difficult to install than UTP cables, but offer better performance in noisy environments.

▼ Straight-through vs crossover

- Straight-through cables are used to connect different types of devices, while crossover cables are used to connect similar types of devices

Straight-through cable:

- Pin 1: White/Orange | Pin 2: Orange | Pin 3: White/Green | Pin 4: Blue
- Pin 5: White/Blue | Pin 6: Green | Pin 7: White/Brown | Pin 8: Brown

Crossover cable:

- Pin 1: White/Green | Pin 2: Green | Pin 3: White/Orange | Pin 4: Blue
- Pin 5: White/Blue | Pin 6: Orange | Pin 7: White/Brown | Pin 8: Brown

2. **Coaxial Cable**: Coaxial cables consist of a copper core, surrounded by insulation, and then a braided metal shield, and finally an outer insulating layer. Coaxial cables are used for cable TV and high-speed internet

connections. They are more expensive than twisted pair cables but can support higher data rates.

3. **Fiber Optic Cable**: Fiber optic cables are used for long-distance communication and can support very high data rates. They consist of a glass or plastic core surrounded by cladding, and then an outer protective layer. Data is transmitted through the core using light signals. Fiber optic cables are immune to electromagnetic interference and can transmit data over long distances without degradation.
4. **Ethernet Cable**: Ethernet cables are used to connect devices in a LAN using the Ethernet protocol. There are several types of Ethernet cables: (1- 5 m ale)
    - Cat 5e: This is the most common type of Ethernet cable and can support speeds up to 1 Gbps.
    - Cat 6: This type of cable can support speeds up to 10 Gbps and is used for high-performance LANs.
    - Cat 7: This type of cable can support speeds up to 100 Gbps and is used for data centers and other high-performance applications.

▼ OSI Layer

> 💡 The OSI (Open Systems Interconnection) model is a conceptual framework that describes how data moves between devices in a network.

1. **Physical** Layer: This layer is responsible for the physical transmission of data between devices. It defines the physical characteristics of the communication medium, such as the voltage levels, cable specifications, and connectors.
2. **Data Link** Layer: This layer is responsible for the reliable transmission of data between adjacent network nodes. It uses techniques such as error detection and correction to ensure that data is transmitted without errors.
3. **Network** Layer: This layer is responsible for the routing of data between different networks. It defines protocols for addressing, routing, and forwarding data packets across multiple networks.

4. **Transport** Layer: This layer is responsible for the reliable transmission of data between end-to-end connections. It ensures that data is transmitted in the correct sequence and without errors.
5. **Session** Layer: This layer is responsible for managing the communication session between two devices. It establishes, maintains, and terminates the connection between devices.
6. **Presentation** Layer: This layer is responsible for the formatting and presentation of data. It defines standards for data representation and conversion, encryption and decryption, and compression and decompression.
7. **Application** Layer: This layer is responsible for providing services to applications that are used by end-users. It defines protocols for email, file transfer, remote login, and other application-level services.

▼ Network addresses

- A network address is a numerical identifier that represents the location of a network on the Internet. It is also known as an IP network address or a network ID.
- An IP address is a unique numerical identifier assigned to every device connected to a network that uses the Internet Protocol for communication. IP addresses are used to route data between devices on a network.
- The network address is determined by the **IP address and subnet mask**.
- Network addresses are used to identify devices on a network. MAC addresses are unique hardware addresses assigned to each network interface card (NIC) and are used at the Data Link layer of the OSI model. IP addresses are logical addresses assigned to devices on a network and are used at the Network layer of the OSI model.
- IP addressing involves assigning IP addresses to devices on a network. IP addresses are 32-bit binary numbers represented in dotted decimal notation (e.g. 192.168.0.1). IP addresses can be static or dynamic, and can be assigned manually or automatically using Dynamic Host Configuration Protocol (DHCP).
- IP addresses are typically divided into two parts: the network ID and the host ID.

> For example, if a device has an IP address of 192.168.1.100 and a subnet mask of 255.255.255.0, then the network address would be 192.168.1.0, since the network ID portion of the IP address is the first three octets (192.168.1) and the subnet mask indicates that the last octet (100) is the host ID.

    - The network ID identifies the network to which the device is connected.
    - The host ID identifies the specific device on that network.
- Subnetting is the process of dividing a larger network into smaller subnetworks, and is used to improve network efficiency and security.
- Subnet Masks:
    - Subnet masks are used to divide an IP address into network and host parts.
    - Subnet masks are represented as a series of 1s followed by a series of 0s. (255.255.255.0 / 255.255.0.0)
    - The number of 1s in the subnet mask determines the number of bits used for the network ID, while the number of 0s determines the number of bits used for the host ID.
    - The number of network IDs and host IDs that can be created with each class of IP address is determined by the number of bits used for the network ID and host ID in the subnet mask.

▼ Network Classes

Classes A, B, and C are used for normal networks, while D and E are used for special cases

- Class A addresses have a subnet mask of 255.0.0.0 and are used for large networks with many devices.
- Class B addresses have a subnet mask of 255.255.0.0 and are used for medium-sized networks.

- Class C addresses have a subnet mask of 255.255.255.0 and are used for small networks.
- Class D: Class D addresses are used for multicasting, which allows a single packet to be sent to multiple destinations at once.
    - For example, video streaming services may use multicasting to distribute live events to multiple viewers simultaneously.
- Class E: Class E addresses are reserved for future use and are not currently used in any public applications. Some sources suggest that they may be used for experimental or research purposes in the future.

▼ IPV4 / IPV6

- IPv4 (Internet Protocol version 4) and IPv6 (Internet Protocol version 6) are two versions of the IP protocol used for communicating between devices on a network.
- IPv4 addresses are 32-bit binary numbers represented in dotted decimal notation (e.g. 192.168.0.1).
- IPv6 addresses are 128-bit binary numbers represented in hexadecimal notation (e.g. 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
- In IPv4 addressing, a network address is obtained by performing a bitwise AND operation between the IP address and the subnet mask. The result of this operation is the network ID, which identifies the specific network to which the device is connected.
- IPv6 was developed to address the limitations of IPv4, which include a limited number of available addresses and security vulnerabilities. IPv6 provides a larger address space, improved security features, and better support for mobile devices and multimedia applications.

▼ TCP / UDP

TCP

- Connection-oriented protocol
- Provides reliable data transmission by establishing a connection between the sender and receiver before data transmission begins
- Uses a three-way handshake to establish a connection

- Guarantees in-order delivery of data
- Provides flow control and congestion control mechanisms to ensure efficient data transmission
- Slower than UDP due to the additional overhead of connection setup and reliability mechanisms
- Commonly used for applications that require reliable and ordered data transmission, such as email, file transfer, and web browsing.

UDP

- Connectionless protocol
- Does not establish a connection before data transmission begins
- Does not guarantee reliable data transmission or in-order delivery of data
- Does not provide flow control or congestion control mechanisms
- Faster than TCP due to the lack of connection setup and reliability mechanisms
- Commonly used for applications that require faster data transmission and can tolerate loss of data or out-of-order delivery, such as video streaming, online gaming, and DNS.

▼ Network Ports

- Network ports are used to establish communication between two devices over a network. Each port is assigned a unique number which helps in identifying the type of service being provided by the device. Some of the most common ports used for communication are:
- Port **20/21** (FTP): This port is used for file transfer between a client and server over the File Transfer Protocol (FTP). Port 20 is used for data transfer, while port 21 is used for control information.
- Port **22** (SSH): Secure Shell (SSH) is a protocol used for secure remote access to devices over a network. Port 22 is used for SSH communication.
- Port **23** (Telnet): Telnet is a protocol used for remote access to devices over a network. Port 23 is used for Telnet communication.

- Port 25 (SMTP): Simple Mail Transfer Protocol (SMTP) is used for email delivery. Port 25 is used for outgoing email communication.
- Port **53** (DNS): Domain Name System (DNS) is used for name resolution of domain names to IP addresses. Port 53 is used for DNS communication.
- Port **80/8080** (HTTP): Hypertext Transfer Protocol (HTTP) is used for communication between web servers and clients. Port 80 is used for HTTP communication, while port 8080 is used for alternative HTTP communication.
- Port 110 (POP3): Post Office Protocol version 3 (POP3) is used for email retrieval. Port 110 is used for incoming email communication.
- Port 143 (IMAP): Internet Message Access Protocol (IMAP) is used for email retrieval. Port 143 is used for incoming email communication.
- Port **443** (HTTPS): Hypertext Transfer Protocol Secure (HTTPS) is a secure version of HTTP used for secure communication between web servers and clients. Port 443 is used for HTTPS communication.
- Port 3389 (RDP): Remote Desktop Protocol (RDP) is used for remote access to Windows-based devices over a network. Port 3389 is used for RDP communication.

▼ Software and information security

▼ Terminologies

1. Authentication - the process of verifying the identity of a user, device, or system.
2. Authorization - the process of determining if a user, device, or system is allowed access to a resource or action.
3. Encryption - the process of converting data into a secret code to prevent unauthorized access during transmission or storage.
4. Decryption - the process of converting encrypted data back to its original form.
5. Firewall - a security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules.

6. Intrusion Detection System (IDS) - a system that monitors network traffic for signs of unauthorized access or malicious activity.
7. Intrusion Prevention System (IPS) - a system that actively blocks malicious traffic from entering a network.
    - Monitors network traffic for signs of unauthorized access, attacks, or other security violations.
    - It can be configured to take automatic actions to prevent attacks.
8. Malware - any software designed to harm a computer system, network, or device.
9. Vulnerability - a weakness or flaw in a system that can be exploited by attackers to gain unauthorized access or cause harm.
10. Patch - a software update designed to fix vulnerabilities or bugs in a system or application.
11. Two-Factor Authentication (2FA) - a security process that requires users to provide two different forms of authentication to verify their identity.
12. Social Engineering - the use of psychological manipulation to trick users into divulging sensitive information or performing certain actions.
13. Advanced Persistent Threat (APT) - a targeted and sophisticated attack that typically involves multiple stages and is designed to gain long-term access to a system or network.
14. Virtual Private Network (VPN): A VPN is a secure network connection that allows remote users to securely access the network over the internet. It provides a secure and encrypted connection between the user and the network.
15. Unified Threat Management (UTM) Device: A UTM device is a network security appliance that combines multiple security functions into a single device, such as firewall, intrusion detection and prevention, antivirus, web filtering, and more.
16. Load Balancer: A load balancer is a network device that distributes incoming network traffic across multiple servers to ensure that no single server is overwhelmed with traffic.
17. Proxy Server: A proxy server is an intermediary server between a user and the internet. It can be used to filter and block access to certain

websites or services.

18. Authentication Server: An authentication server is a device that provides centralized authentication and authorization services for network users.

Introduction

- Goals of security: CIA triad - confidentiality, integrity, availability.

Threats and Attacks

- **Threat**: a potential violation of security.
    - Types of threats: **passive** (eavesdropping), **active** (modification, **injection**, **denial of service).**
- **Risk:** A measure of the extent to which an entity is threatened by a potential circumstance or event
    - the adverse impacts that would arise if the circumstance or event occurs
    - the likelihood of occurrence.
- **Exploit**: a code that takes advantage of a software vulnerability or security flaw
- **Attack**: an actual violation of security.
- Types of attacks: reconnaissance, access, denial of service, etc.
    - **Denial of Service (DoS)** - an attack that floods a network or system with traffic, rendering it unavailable to legitimate users.
    - **Distributed Denial of Service (DDoS)** - an attack in which multiple compromised devices are used to flood a network or system with traffic, making it unavailable to legitimate users.
    - **Virus** - a malicious program that replicates itself by attaching to a clean file and infecting other files on the system.
    - **Worm** - a type of malware that spreads across a network by exploiting security vulnerabilities in computers.
    - **Trojan** - a program that appears to be legitimate but is actually designed to perform malicious actions on a system.
    - **Ransomware** - a type of malware that encrypts files on a system and demands payment in exchange for the decryption key.

- **Botnet** - a network of compromised computers that are controlled by a remote attacker to perform malicious actions, such as sending spam or launching DDoS attacks.
- **Spoofing** - a technique in which an attacker impersonates a legitimate source, such as an email address or IP address, to gain unauthorized access to a system or network.
- **Phishing** - a type of social engineering attack in which attackers use deceptive emails or messages to trick users into divulging sensitive information.
- **Zero-Day Exploit** - a security vulnerability that is unknown to the system or application developer, giving attackers the advantage of exploiting it before a patch is released.
- **Man-in-the-Middle (MitM)** - an attack in which an attacker intercepts communication between two parties and can potentially eavesdrop, manipulate, or steal data.

Cryptography

- Cryptography: the science of hiding messages.
- Symmetric key encryption: same key used for encryption and decryption.
- Asymmetric key encryption: public key used for encryption, private key used for decryption.
- Hashing: one-way encryption.

Access Control

- Access control: the process of determining who is authorized to access what.
- Types of access control: discretionary, mandatory, role-based.
- Authorization: granting access to a specific resource.
- Authentication: verifying the identity of a user.

Software Security

- Security in the software development lifecycle.
- Input validation: ensuring that input data is valid and safe.

- Buffer overflows: when a program tries to store more data in a buffer than it was intended to hold.
- Injection attacks: injection of malicious code.
- Cross-site scripting: a type of injection attack where an attacker injects malicious code into a web page.
- Cross-site request forgery: tricking a user into performing an action they didn't intend to.
- Security testing: testing to identify vulnerabilities.

Web Security

- Web security: securing web applications and web services.
- OWASP Top 10: the top 10 web application vulnerabilities.
- Secure Sockets Layer (SSL): a protocol for secure communication over the internet.
- Transport Layer Security (TLS): a successor to SSL.

Network Security

- Network security: securing networks and network devices.

▼ Fundamentals of Software Engineering

- Software Development Life Cycle (SDLC): The SDLC is a process used to develop software applications from conception to retirement.
- Consists of phases such as requirements gathering, design, implementation, testing, deployment, and maintenance.
- Software Design: It includes decisions about architecture, data structures, algorithms, and interfaces.
- Software Configuration Management (SCM): SCM is the process of managing changes to software code and other artifacts throughout the SDLC. It includes version control, change management, and release management.
- Software Quality: Quality refers to how well the software meets its requirements and performs its intended functions. It can be measured

through various metrics such as defect density, code coverage, and user satisfaction.

- Software Project Management: Project management involves planning, organizing, and controlling resources to achieve specific software development goals. It includes tasks such as scheduling, budgeting, risk management, and stakeholder management.

Different SDLCs

1. **Waterfall Model:** This is a sequential approach, where each stage is completed before moving on to the next. It includes stages like requirements gathering, design, development, testing, and maintenance.
2. **Agile Model**: This is an iterative approach that involves continuous collaboration and feedback between developers, testers, and customers. It includes stages like planning, designing, developing, testing, and releasing small increments of the software.
3. Iterative Model: This model is similar to the Agile model in that it involves multiple iterations. However, the iterations are larger and longer, and the end goal is to deliver a complete product at the end of each iteration.
4. **Spiral Model:** This model combines the iterative approach of the Iterative Model with the risk assessment of the Waterfall Model. It involves creating a prototype, testing it, and then evaluating the risks before moving on to the next stage.
5. **V-Model:** This model is similar to the Waterfall Model, but includes testing at each stage to ensure that the requirements are being met. It includes stages like requirements gathering, design, testing, and maintenance.
6. **Big Bang Model:** This model involves developing the entire system at once, without any clear plan or structure. It is not widely used, as it is considered risky and can lead to many errors and problems.

▼ Requirement Engineering, Architecture and Design

▼ Requirement engineering

- Requirement Engineering is the process of identifying, analyzing, documenting, validating, and managing the requirements of a software system.

    - Lays the foundation for the entire software development process.
- The goal of Requirement Engineering is to produce a complete, consistent, and unambiguous specification of the software requirements. This specification should provide a clear understanding of what the system needs to do, how it needs to behave, and what constraints it needs to operate under.

Steps involved in the Requirement Engineering process:

1. Requirements Elicitation: gathering requirements from various stakeholders, such as customers, users, and domain experts.
2. Requirements Analysis: This involves analyzing and prioritizing the requirements to determine their feasibility and importance. This step also involves resolving any conflicts or ambiguities in the requirements.
3. Requirements Specification: This involves documenting the requirements in a clear and unambiguous manner. This can be done using natural language, diagrams, or formal notations such as UML.
4. Requirements Validation: This involves reviewing the requirements to ensure that they are complete, consistent, and correct. This can be done through walkthroughs, inspections, and testing.
5. Requirements Management: This involves maintaining and updating the requirements throughout the software development process. This includes tracking changes to the requirements, and ensuring that they are still relevant and valid.

Some of the techniques commonly used in Requirement Engineering include:

- Interviews, Use Cases, Prototyping, Formal Methods, Requirements Management Tools

▼ Architecture and Design

Architecture Styles

- Client-Server: where the system is divided into clients and servers, with the clients requesting services from the servers.
- N-tier: where the system is divided into multiple layers, each responsible for a specific task.

- Micro-services: where the system is divided into small, independent services that work together to achieve the system's goals.
- Event-Driven: where the system is designed to respond to events generated by other systems or users.

Architecture Patterns

- Model-View-Controller (MVC): where the system is divided into three components - model, view, and controller - to separate the concerns of data, presentation, and control logic.
- Repository Pattern: where the system is designed to interact with a data repository to perform data access and manipulation operations.
- Dependency Injection (DI): where the system is designed to inject dependencies into objects rather than having the objects create their own dependencies.
- Service-Oriented Architecture (SOA): where the system is designed to provide services that can be consumed by other systems.

Architectural Views

- **Logical view**: shows the system's functionality and how it is organized.
- **Process view**: shows how the system processes and communicates data.
- **Physical view**: shows the system's physical components and how they are connected.
- **Development view**: shows how the system is built and how its components are developed and integrated.

Performance Considerations

1. Performance testing: Performance testing is a type of software testing that is used to evaluate the speed, responsiveness, stability, and scalability of a software application or system under different workload conditions. The goal of performance testing is to identify performance bottlenecks and ensure that the system can handle the expected user load.

2. Caching: Caching is a technique used to improve the performance and speed of software applications by storing frequently accessed data or information in a temporary storage area. This allows the application to access the data quickly and efficiently, without having to fetch it from the original source every time it is needed.
3. Load balancing: Load balancing is the process of distributing network traffic across multiple servers or resources to optimize resource utilization, maximize throughput, and minimize response time. Load balancers are used to manage the workload of a software system by distributing requests to different servers, which helps to improve the overall performance and reliability of the system.
4. Scaling: Scaling refers to the process of increasing the capacity or capability of a software system to handle a growing number of users, requests, or data. There are two types of scaling: vertical scaling and horizontal scaling. Vertical scaling involves adding more resources to a single server or machine, while horizontal scaling involves adding more servers or resources to the system.
5. Optimization: Optimization refers to the process of improving the performance and efficiency of a software application or system by identifying and eliminating bottlenecks, reducing resource usage, and improving response times. Optimization can involve a range of techniques, including code optimization, database optimization, and system tuning. The goal of optimization is to improve the user experience, reduce costs, and increase the scalability and reliability of the system.

▼ Design pattern

Design patterns are reusable solutions to commonly occurring design problems. There are several design patterns that can be used in software architecture.

1. **Creational patterns:** These patterns deal with object creation mechanisms, trying to create objects in a manner suitable to the situation. Some examples include Singleton, Factory, and Builder.
2. **Structural patterns:** These patterns deal with object composition, creating relationships between objects to form larger structures. Some examples include Adapter, Facade, and Decorator.

3. **Behavioral patterns:** These patterns deal with communication between objects, focusing on the way objects collaborate to achieve complex behaviors. Some examples include Observer, Command, and Strategy.

- Creational patterns
  - Singleton pattern
  - Factory pattern
  - Abstract Factory pattern
  - Builder pattern
  - Prototype pattern
- Structural patterns
  - Adapter pattern
  - Bridge pattern
  - Composite pattern
  - Decorator pattern
  - Facade pattern
  - Flyweight pattern
  - Proxy pattern
- Behavioral patterns
  - Chain of Responsibility pattern
  - Command pattern
  - Interpreter pattern
  - Iterator pattern
  - Mediator pattern
  - Memento pattern
  - Observer pattern
  - State pattern
  - Strategy pattern
  - Template Method pattern
  - Visitor pattern

Uses

- Improved code maintainability: easier to modify and maintain code because they provide a standard set of solutions to common problems.
- Better code quality: cleaner, more modular, and more flexible code.

- Code reusability: create reusable code, reducing development time and improving efficiency.

▼ Software Project Management

1. Project Planning: This involves defining the scope of the project, identifying the resources required, developing a timeline and schedule, and estimating the budget and cost of the project.
2. Risk Management: This involves identifying potential risks to the project, assessing the likelihood and impact of these risks, and developing a plan to mitigate or avoid them.
3. Resource Management: This involves managing the people, tools, and equipment required to complete the project. It also involves assigning roles and responsibilities to team members and managing the team's workload.
4. Quality Management: This involves ensuring that the software being developed meets the requirements and standards set out for the project. It involves testing, reviewing, and verifying the software throughout the development process.
5. Change Management: This involves managing changes to the project scope, timeline, or budget. It involves assessing the impact of changes and making decisions about whether to proceed with them or not.
6. Communication Management: This involves ensuring that everyone involved in the project is informed about its progress, changes, and issues. It involves establishing regular communication channels and holding regular meetings to discuss project status.
7. Project Monitoring and Control: This involves tracking project progress against the plan, identifying deviations from the plan, and taking corrective action to keep the project on track.
8. Project Closure: This involves completing the project and closing it out. It involves documenting lessons learned, archiving project materials, and evaluating the success of the project.

▼ Software Testing, Verification and Quality Assurance

- Testing is an integral part of the software development life cycle (SDLC) and helps to identify defects and errors early in the process, which saves time, effort, and cost in the long run.
- **Verification** is focused on making sure system-level requirements are met, while
- **Validation** focuses on making sure the system is performing the way it was designed and intended

Testing

1. **Unit Testing**: This is a testing method that tests individual software components, such as functions or methods, to ensure that they are working as expected.
2. **Integration Testing**: This method tests how different software components work together as a group, to ensure that they integrate smoothly and without conflicts.
3. **System Testing:** This method tests the entire software system as a whole, to ensure that it meets the specified requirements and performs the intended functions.
4. **Acceptance Testing:** This method involves testing the software with a set of predefined acceptance criteria, to ensure that it meets the customer's expectations and requirements.
5. **Regression Testing:** This method involves retesting previously tested software after making changes to ensure that the changes did not affect the software's performance or functionality.
6. **Performance Testing:** This method tests the software's performance under various conditions, such as high traffic or stress, to ensure that it can handle the expected load.
7. **Security Testing:** This method tests the software's security features, such as authentication and authorization, to ensure that it is secure from unauthorized access.

Testing techniques

1. **Black-box Testing**: This technique tests the software without knowing its internal workings, to ensure that it meets the specified requirements.

2. **White-box Testing**: This technique tests the software's internal workings, such as code and logic, to ensure that it is working as intended.
3. **Grey-box Testing**: This technique is a combination of black-box and white-box testing, where the tester has some knowledge of the software's internal workings.

Quality Assurance (QA)

- The QA process involves defining and implementing a set of standards and procedures that ensure the software meets the required quality standards.
- Defining quality standards, Test planning, Test execution, Defect tracking and management, Continuous improvement

▼ Operating System and System Programming [Not composed well]

▼ Operating system [incomplete]

▼ General

- An operating system (OS) is a software system that manages computer hardware and software resources and provides common services for computer programs.
- An operating system is a crucial component of any computer system, and performs several important functions such as process management, memory management, file management, device management, and security.
- There are several types of operating systems, each designed for specific applications and use cases.

▼ CPU cycles

- CPU cycles, also known as clock cycles, are the basic unit of execution time in a computer's central processing unit (CPU).
- Each CPU cycle consists of a sequence of steps that are used to execute a single instruction.

> 💡 The steps involved in a CPU cycle can vary depending on the architecture of the CPU, but typically include **fetching** the instruction from memory, **decoding** the instruction, **executing** the instruction, and **storing** the results back in memory.

- The clock speed of a CPU is measured in hertz (Hz) and determines how many cycles the CPU can execute per second. For example, a CPU with a clock speed of 3.0 GHz can execute 3 billion cycles per second.
- The efficiency of a CPU can be improved by reducing the number of cycles required to execute a given set of instructions.

▼ Scheduling

- Scheduling is the process of determining which process should run on a CPU at a given time.
- Operating systems use scheduling algorithms to manage the execution of processes and to allocate resources fairly and efficiently.

  There are several different types of scheduling algorithms:

  1. **First-Come, First-Served (FCFS):** This is the simplest scheduling algorithm, where the CPU is given to the first process that arrives and waits for its turn.
  2. **Shortest Job First (SJF):** This algorithm gives the CPU to the process with the shortest burst time (i.e., the amount of time it takes to complete).
  3. **Priority Scheduling:** This algorithm assigns priorities to processes and gives the CPU to the process with the highest priority.
  4. **Round Robin (RR):** This algorithm assigns a time slice (a fixed amount of time) to each process in turn, and rotates the CPU among the processes in a circular fashion.
  5. **Multilevel Feedback Queue (MFQ):** This algorithm assigns processes to different queues based on their priority and time requirements. Processes in higher priority queues are given CPU

time first, and processes in lower priority queues are given CPU time only when higher priority queues are empty.

▼ System programming

▼ Introduction to System Programming

- System programming involves writing low-level software that interacts with the operating system and hardware of a computer system.
- It is used to build system-level software like device drivers, operating systems, and network protocol stacks.
- The goal of system programming is to create efficient and robust software that can run on a wide range of computer hardware and operating systems.

▼ Operating System Basics

- The operating system is responsible for managing system resources like memory, CPU, and input/output devices.
- System calls are used to request services from the operating system like file I/O, process management, and network communication.
- Process scheduling is the process of assigning system resources to running programs.
- Memory management involves managing the allocation and deallocation of memory to running programs.

▼ Interprocess Communication

- Interprocess communication (IPC) is the mechanism by which processes can communicate with each other.
- IPC mechanisms include shared memory, message queues, pipes, and sockets.
- IPC is used for tasks like synchronization between processes and sharing data between processes.

▼ Concurrency

- Concurrency is the ability of a system to execute multiple tasks at the same time.

- Threads are a way to achieve concurrency within a process.
- Synchronization mechanisms like locks and semaphores are used to prevent race conditions and ensure correct behavior in concurrent systems.

▼ Debugging and Profiling:

- Debugging is the process of finding and fixing errors in software.
- Debugging tools like gdb and printf statements are used to locate errors.
- Profiling is the process of measuring the performance of software.
- Profiling tools like valgrind and gprof are used to identify performance bottlenecks and optimize code.

▼ System-Level Programming Languages

- System-level programming languages like C and Assembly are commonly used in system programming.
- These languages provide low-level control over system resources and allow for efficient system programming.
- Higher-level programming languages like Python and Java are also used in system programming, but are less common.

▼ Stages of compiling a C program

- **Preprocessing**: Removal of Comments, Expansion of Macros, Expansion of the included files, Conditional compilation, outputs .i file
- **Compiling**: produce an; intermediate compiled output file **filename.s.,** This file is in assembly level instructions.
- **Assembler**: he filename.s is taken as input and turned into **filename.o**
    - existing code is converted into machine language, the function calls like printf() are not resolved.
- **Linker** : linking of function calls with their definitions are done

▼ Artificial intelligence

- **Supervised** learning, the algorithm is trained on a dataset of labeled examples to make predictions on new, unseen data.
- **Unsupervised** learning, the algorithm tries to find patterns or structure in a dataset without explicit labeling.
- **Reinforcement** learning involves learning through a system of rewards and punishments.

▼ Big data Modeling

- Big Data Modeling involves designing and implementing structures to store, process, and analyze large volumes of structured and unstructured data.

1. Data Modeling: This involves creating a logical and physical data model that describes the relationships between data entities and attributes. The logical data model describes the data in terms of its business meaning, while the physical data model describes how the data is stored in the database.
2. Data Storage Technologies: There are various technologies for storing and managing big data. The choice of technology depends on the data requirements, performance needs, and scalability of the system.
3. Data Processing Workflows: Big data processing involves data ingestion, data processing, and data analysis.
4. Data Integration: Big data often comes from multiple sources, and integrating the data requires tools that can handle data from diverse sources.
    - Data integration involves extracting data from various sources, transforming the data to a common format, and loading the data into the target system.
5. Data Governance: Ensure that the data is accurate, consistent, and meets regulatory requirements. This involves defining data policies, roles, and responsibilities, and implementing data quality checks.
6. Data Security: This includes access control, authentication, encryption, and audit trails to ensure that data is only accessible to authorized personnel.
7. Performance Optimization: This involves tuning the system parameters, optimizing the data processing workflows, and selecting appropriate hardware for the system.

▼ Java

▼ Basics

- Java is compiled into bytecode, which can be executed on any platform with the help of the Java Virtual Machine (JVM).
- The syntax of Java is similar to that of C++, but it eliminates some of the more complex features of C++ such as pointers and operator overloading.
- Java supports automatic garbage collection, which means that it automatically manages memory allocation and deallocation.
- JVM stands for Java Virtual Machine. It is a virtual machine that runs the compiled Java code, which is converted to bytecode. The JVM interprets the bytecode and translates it into machine code, which can then be executed on the host operating system. The JVM provides a platform-independent execution environment for Java programs.
- JRE stands for Java Runtime Environment. It includes the JVM, class libraries, and other components necessary to run Java applications. The JRE is a subset of the JDK (Java Development Kit), which includes the tools needed to develop and compile Java code.
- The JVM is responsible for executing Java programs, while the JRE provides the environment in which the JVM runs.
- The JVM is available for various platforms, including Windows, Linux, and Mac OS. This allows Java programs to be executed on different operating systems without the need for platform-specific modifications.
- The JRE includes the Java class libraries, which provide a set of pre-written classes and interfaces that can be used to develop Java applications. These libraries include classes for working with input/output, networking, data structures, and more.
- The JRE also includes the Java Development Kit (JDK), which provides tools for developing, compiling, and debugging Java code. The JDK includes the Java compiler, which converts Java code to bytecode that can be executed by the JVM.
- The JRE is required to run Java applications on a user's computer. It can be downloaded and installed separately from the JDK.
- In summary, the JVM provides the platform-independent execution environment for Java programs, while the JRE includes the JVM, class libraries, and other components necessary to run Java applications.

▼ Exception handling

- Exceptions are thrown using the throw keyword and caught using try-catch blocks.
- Finally blocks can be used to ensure that certain code is executed regardless of whether an exception is thrown or not.

▼ Multithreading

- Java supports multithreading, which allows for the concurrent execution of multiple threads of execution.
- Threads can be created using the Thread class, which provides methods for starting, stopping, and controlling the execution of threads.
- Synchronization mechanisms such as locks and semaphores can be used to ensure that multiple threads can access shared resources without interfering with each other.

▼ Input/output

- Input can be read using the Scanner class, which provides methods for reading input from the console and other sources.
- Output can be written using the System.out.println() method, which writes to the console, or using the FileWriter class, which allows for writing to files.

▼ Collections

- Java provides a wide range of collection classes, to store and manipulate groups of objects.
- The most common collection classes are ArrayList, LinkedList, HashMap, and HashSet.
- Collections can be iterated using loops such as for-each loops or using the Iterator interface.

▼ SQL

SQL: Structured Query Language (SQL) is the standard language used for querying and manipulating data in relational databases.

- SELECT: used to retrieve data from a table

  Example: `SELECT * FROM customers WHERE last_name = 'Smith';`

- INSERT: used to add data to a table

  Example: `INSERT INTO customers (first_name, last_name, email) VALUES ('John', 'Doe', 'johndoe@email.com');`

- UPDATE: used to modify existing data in a table

  Example: `UPDATE customers SET email = 'newemail@email.com' WHERE customer_id = 123;`

- DELETE: used to remove data from a table

  Example: `DELETE FROM customers WHERE customer_id = 123;`

- CREATE: used to create a new table or database structure

  Example: `CREATE TABLE orders (order_id INT, customer_id INT, order_date DATE, total_cost DECIMAL(10,2));`

- ALTER: used to modify the structure of an existing table

  Example: `ALTER TABLE customers ADD COLUMN age INT;`

- DROP: used to delete an entire table or database structure

  Example: `DROP TABLE customers;`

Joins

• Joins are used in SQL to combine data from two or more tables based on a related column between them.

- **Inner join:** returns only the matching rows between two tables based on the related column.

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

- **Left join**: returns all the rows from the left table and the matching rows from the right table based on the related column. If there is no match in the right table, the result will be a NULL value for the right table columns.

```
SELECT column_name(s)
FROM table1
```

```
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

- **Right join**: returns all the rows from the right table and the matching rows from the left table based on the related column. If there is no match in the left table, the result will be a NULL value for the left table columns.

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

- **Full outer join:** returns all the rows from both tables and NULL values if there is no match in the related column.

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name;
```