1. What is a Program?
   A program can be defined as a set of Instructions.

2. What is Programming??
   Creating a program can be defined as Programming or how you get computers to solve problems.

3. What are the key steps involved in the programming process?

4. Types of Programming Language?
   Types of programming language are as follows:
   - ➢ Low-level programming language.
   - ➢ High-level programming language.

5. What are the main differences between high-level and low-level programming languages?
   A high-level language is a user-friendly language designed to develop user- friendly software programs and websites whereas, a low-level language is machine dependent (0’s and 1’s).

6. World Wide Web & How Internet Works?
   WWW or World Wide Web commonly known as Web is a cluster of websites of web pages stored in web severs. These usually contains text pages, images, videos and so on. This can we accessed through devices like laptops, desktops, Smartphones etc.
   \>>>>>how internet works<<<<<

7. --
8. Software and its types?
   Software can be defined as a set of instructions. Types of software include;
   - ➔ Application Software
   - ➔ System Software
   - ➔ Driver Software
   - ➔ Middleware
   - ➔ Programming Software.

9. Identify and classify 5 applications you use daily as either system software or application software?

SYSTEM SOFTWARE	APPLICATION SOFTWARE
MS OFFICE	INSTAGRAM
CAMERA	WHATSAPP
CALCULATOR	FACEBOOK

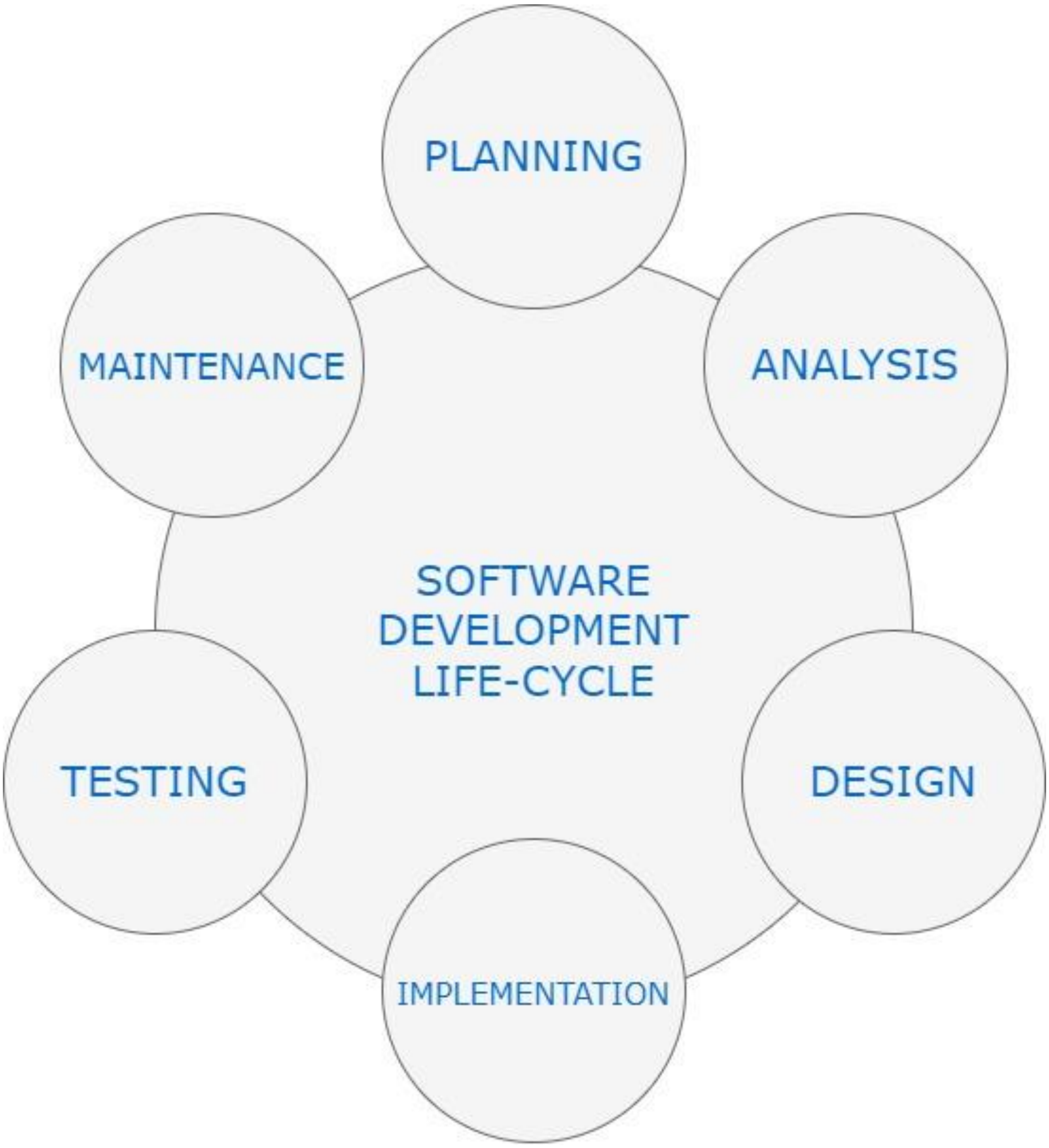
NOTEPAD	LINKEDIN
WINDOWS/ANDROID/IOS	NETFLIX

10. What is the difference between system software and application software?

SYSTEM SOFTWARE	APPLICATION SOFTWARE
Designed to run a computer’s application programs and hardware.	Designed for a user-friendly interface and different high-level languages are used to build application software.

11. ---

12.Create a flowchart representing the Software Development Life Cycle (SDLC)?



13.: What are the main stages of the software development process?

➔ PLANNING

➔ ANALYSIS

➔ DESIGNING

➔ IMPLEMENTATION

➔ TESTING

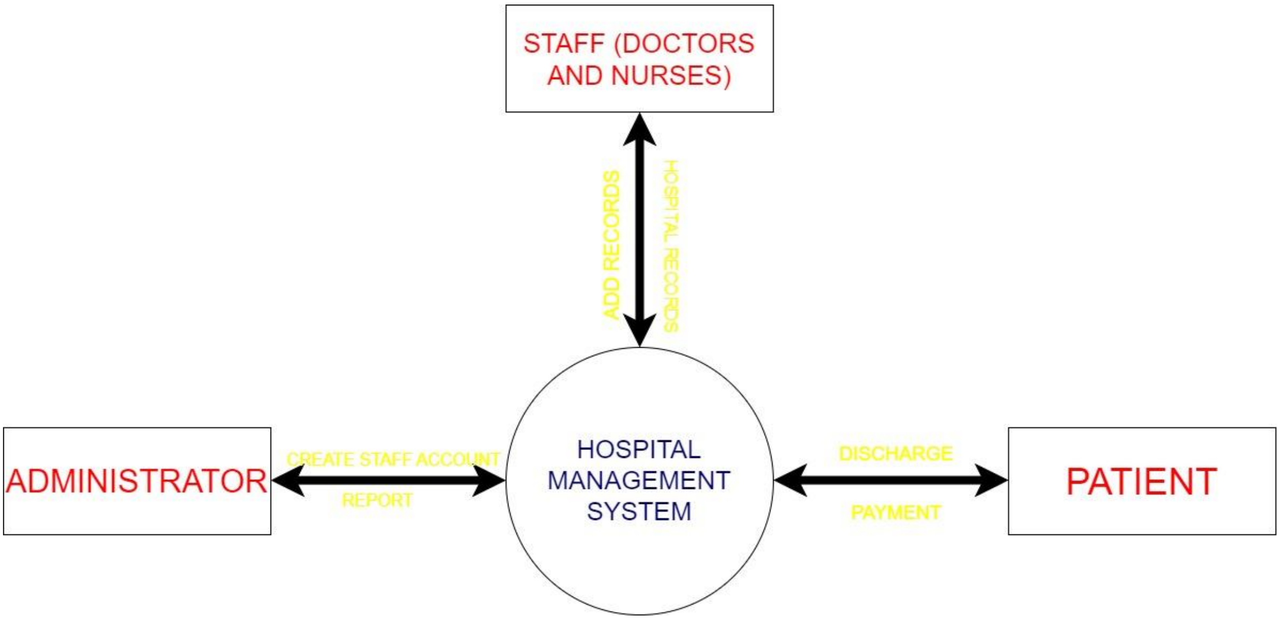
➔ MAINTENANCE

1. DFD

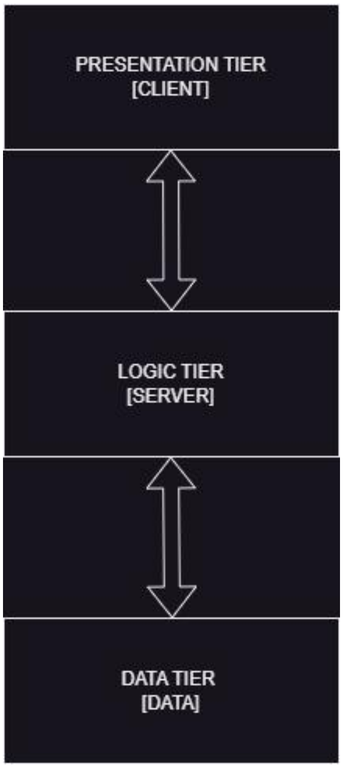
   Data Flow Diagram or DFD is a flow of data of a system or a representation of a process. It gives details about the inputs and outputs of each entity and the process itself. Components of DFD comprises of :

   ➔ Process.
   ➔ Data Flow.
   ➔ Data Store.
   ➔ External Entity.

2. Create a DFD for a hospital management system.?



3. Design a basic three-tier software architecture diagram for a web application.



4. Why are layers important in software architecture?
   The layers in software architecture helps to make the application easier to manage and maintain. It helps to promote modularity and separation of concerns, making it easier for teams to work independently without stepping on each other's toes.

5. Create a case study on the functionality of the presentation, business logic and data access layers of a given software system?

1. Presentation Layer

Description:

The presentation layer serves as the user interface of the system. It interacts with end-users through a web or mobile application.

Key Features:

- **User Interface (UI)**: Built using React for web and Swift for iOS applications.
- **Functionalities**:
  - Product Search and Filters
  - Shopping Cart Management
  - Checkout and Payment
  - User Login and Registration

Responsibilities:

- Captures user input (e.g., login credentials, product selections).
- Displays processed data from the Business Logic Layer, such as available products or order confirmations.
- Ensures responsiveness and accessibility for a wide range of devices.

Example Interaction:

A customer searches for "wireless headphones." The search query is passed to the Business Logic Layer via an API.

2. Business Logic Layer

Description:

The business logic layer contains the core application rules and logic. It acts as a bridge between the Presentation Layer and the Data Access Layer.

Key Features:

- **Validation**: Ensures user input is correct and secure (e.g., valid payment details).
- **Processing**: Handles complex operations like inventory updates, discount calculations, and order validation.
- **Integration**: Communicates with external services, such as payment gateways or shipping APIs.

Responsibilities:

- Coordinates workflow between the Presentation and Data Access Layers.
- Applies business rules, such as promotional discounts or loyalty points.

- Ensures security through input sanitization and encryption during sensitive operations.

  Example Interaction:

  The query for "wireless headphones" is processed to retrieve matching products based on availability and user preferences. The results are passed back to the Presentation Layer for display.

3. Data Access Layer

Description:

The Data Access Layer interacts directly with the database to perform CRUD (Create, Read, Update, Delete) operations. It abstracts data storage details from the Business Logic Layer.

Key Features:

- **Database Operations**: Uses SQL for relational data and NoSQL for unstructured data.
- **Optimization**: Implements caching and query optimization to ensure high performance.
- **Security**: Ensures data integrity and security through encryption and access controls.

Responsibilities:

- Retrieves and stores data efficiently.
- Manages connections to the database and external data sources.
- Translates business logic queries into database operations.

Example Interaction:

The processed search query requests data on "wireless headphones" from the database. The Data Access Layer retrieves matching records and forwards them to the Business Logic Layer.

6. Explore different types of software environments (developments, testing, production). Set up a basic environment in a virtual machine.

1. Development Environment

- **Purpose**: Used by developers to write and debug code.
- **Characteristics**:
  - Contains tools like IDEs (e.g., Visual Studio Code, IntelliJ), compilers, and debuggers.
  - Frequent changes as developers build and modify code.
  - Local databases or mock services often used for rapid testing.
- **Typical Setup**:
  - IDE
  - Version control system (e.g., Git)
  - Local instances of dependencies (e.g., databases, APIs).

2. Testing Environment

- **Purpose**: Validates the functionality of the software before releasing it to production.
- **Characteristics**:
    - Mimics the production environment as closely as possible.
    - Testers use it to run unit tests, integration tests, and user acceptance tests (UAT).
    - Automated and manual testing tools are used.
- **Typical Setup**:
    - Test automation frameworks (e.g., Selenium, JUnit).
    - Staging servers and databases that reflect production configurations.
    - Monitoring tools to detect bugs or performance issues.

3. Production Environment

- **Purpose**: Hosts the live application accessed by end-users.
- **Characteristics**:
    - Stability, security, and performance are critical.
    - High availability, redundancy, and monitoring systems in place.
    - Minimal changes are made directly to this environment.
- **Typical Setup**:
    - Cloud hosting (e.g., AWS, Azure) or dedicated servers.
    - Load balancers and auto-scaling configurations.
    - Logging and monitoring tools like Prometheus and Grafana.

Setting up a basic environment in a virtual machine.

- A virtual machine tool like VirtualBox or VMware.
- An operating system image (e.g., Ubuntu 22.04 ISO).
- Basic understanding of terminal commands.

7. What is the difference between source code and machine code?
   The primary difference between **source code** and **machine code** lies in their purpose, representation, and readability

8. Why is version control important in software development?
   Version control is a critical component of modern software development because it provides a structured way to manage changes to source code over time. It is essential for collaboration, organization, and ensuring the integrity of a project.

9. What are benefits of using Github for students?
   GitHub offers numerous benefits for students, empowering them to learn, collaborate, and showcase their skills effectively.

10. Why is software testing important?

Software testing is a crucial aspect of the software development lifecycle. It ensures that the application meets its requirements, functions correctly, and delivers a high-quality user experience.

It is essential for delivering a product that is reliable, secure, and aligned with user and business needs. It mitigates risks, enhances user satisfaction, and ensures a smooth, trouble-free deployment. Testing isn't just a phase in development—it's a continuous process that supports the long-term success of the software.

11. What types of software maintenance are there?

Software maintenance is a critical aspect of the software lifecycle that ensures the software remains functional, secure, and relevant.

- **Longevity**: Ensures the software remains useful and relevant over time.
- **User Satisfaction**: Keeps the software aligned with user needs and expectations.
- **Security**: Addresses vulnerabilities to protect data and systems.
- **Efficiency**: Optimizes performance and reduces long-term costs.

12. What role does UI/UX design play in application development?

UI (User Interface) and UX (User Experience) design are crucial components of application development. They ensure that the application is not only functional but also user-friendly, visually appealing, and aligned with user needs.

UI and UX design play complementary roles in application development. UI focuses on the look and feel, while UX emphasizes the experience and functionality. Together, they:

- Enhance user satisfaction.
- Drive engagement and retention.
- Support business objectives.
- Improve development efficiency.

By prioritizing UI/UX design, developers and stakeholders can ensure the application is not only functional but also a delight to use.

13. What is difference between native and hybrid mobile apps?

Native and hybrid mobile apps differ significantly in how they are built, their performance, and their development approach.

-> **Choose Native Apps** if:

- You need high performance, seamless user experience, or full device feature access.
- The project is long-term with a larger budget.

-> **Choose Hybrid Apps** if:

- You need a faster, cost-effective solution that works across multiple platforms.
- The app has simpler functionalities and isn't heavily reliant on platform-specific features.

14. What are the pros and cons of desktop applications compared to web applications?

➔ **Choose Desktop Applications** if:

- You need high performance and offline access.
- You require full control over system resources.
- The app is complex or resource-intensive (e.g., games, video editing).

➔ **Choose Web Applications** if:

- You want cross-platform accessibility and remote access.
- You need to collaborate in real time.
- Your app requires frequent updates or doesn't require intensive hardware resources.

15. Research and create a diagram of how data is transmitted from a client to a server over the internet.

To understand how data is transmitted from a client (such as a web browser) to a server over the internet, it's helpful to break it down step by step. Here's a simplified diagram of the process, which follows the key stages of packet transmission:

1. **Client Request**: The process begins when the user inputs a URL (e.g., "www.example.com") into their browser. This triggers a **HTTP request** to the server hosting the website.
2. **DNS Resolution**: The domain name of the website is resolved into an IP address using the **Domain Name System (DNS)**, which tells the client where to send the request.
3. **Packetization**: The data is then divided into **packets**, each with a header containing routing information, such as the source and destination IP addresses. The **Transmission Control Protocol (TCP)** ensures these packets are sent in order, and **Internet Protocol (IP)** determines how they are routed.
4. **Routing through Network**: The packets travel through various **routers** on the internet. These routers direct the packets towards the server's IP address, using a method called **packet switching**. This allows packets to take different routes based on available paths and network congestion

   .
5. **Server Reception**: Once the packets reach the server, the **server's network interface card (NIC)** receives them. The **TCP layer** reassembles the packets into the original request data and forwards it to the **HTTP application layer**.
6. **Response and Acknowledgement**: The server processes the request (e.g., generating a webpage) and sends back a response, which follows the same path in reverse. This response is also packetized, sent back to the client, and reassembled in the browser for display

16. Describe the roles of the client and server in web communication.

In web communication, **the client** and **the server** play distinct but interconnected roles. Here's a breakdown of their functions:

The Client

The client refers to the device or software (e.g., a web browser, mobile app, or desktop application) that requests and receives information over the internet. The key roles of the client in web communication include:

1. **Initiating Requests**:
   - The client initiates the communication process by sending requests to a server. For example, when a user types a URL into a browser, the browser (client) sends a request to the server hosting the website.
2. **Rendering Data**:
   - After receiving the data from the server, the client is responsible for displaying the content. For web browsers, this means rendering HTML, CSS, and JavaScript to present the web page to the user.
3. **User Interaction**:
   - The client interacts with the user, accepting input (e.g., forms, clicks) and sending that information back to the server. It can also handle local processing of data (e.g., executing JavaScript) before sending requests or displaying the response.
4. **Session Management**:
   - The client may also handle session management, like storing cookies or session tokens to maintain state across multiple requests.

The Server

The server is a remote computer that stores and serves resources requested by the client. It plays several critical roles in the communication process:

1. **Receiving and Processing Requests**:
   - When the client sends a request (usually through HTTP/HTTPS), the server receives it and processes it. This could involve querying a database, reading files, or performing computations based on the request.
2. **Sending Responses**:
   - After processing the client's request, the server sends a response back to the client. This could be the requested HTML page, a JSON object, an image, or an error message, depending on the request.
3. **Data Storage**:
   - Servers store the data required by the client, whether it's static (like images or HTML pages) or dynamic content (like user data or content stored in databases).
4. **Security and Validation**:
   - The server is also responsible for validating requests, ensuring that the client has the right permissions to access certain data, and protecting against malicious actions (e.g., through encryption, firewalls, or input validation).

17. Explain the function of the TCP/IP model and its layers.

The TCP/IP model is essential for enabling the vast and diverse communication infrastructure of the internet. By organizing communication into layers, each responsible for specific tasks, the model allows for flexible and reliable data transmission across networks. It is foundational to many internet-based protocols and services, from web browsing to email and file transfer.

18. How does broadband differ from fiber-optic internet?
 Broadband refers to a broad range of internet connections that offer high speeds, **fiber-optic internet** is a specific type of broadband that uses cutting-edge technology for ultra-fast speeds, low latency, and exceptional reliability.

19. What is the role of encryption in securing applications?
Encryption plays a crucial role in securing applications by ensuring that sensitive data remains private and protected from unauthorized access during transmission or while stored. It uses algorithms to transform plaintext data into unreadable ciphertext, which can only be deciphered by someone who has the proper decryption key.