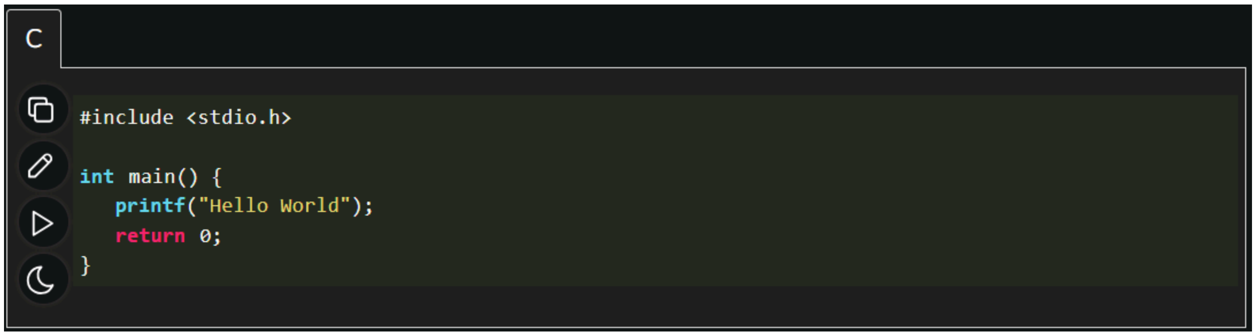
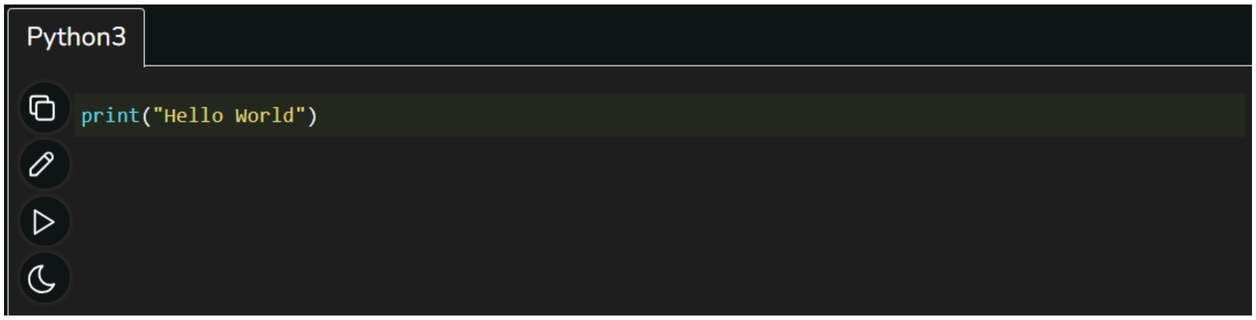
Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.



```c
#include <stdio.h>

int main() {
   printf("Hello World");
   return 0;
}
```

This is “ hello world “ in C language



```python
print("Hello World")
```

This is “ hello world “ in python language.

After observing this codes . i can observe that in c language the coding is bit long . more line or a longer code is required to get the output . but in python language i can see that we have to write fewer lines to get the same out put. Python language is more easy to write codes

Q2 Explain in your own words what a program is and how it functions. What is Programming?

Programming is a process in which we give the computer a set of instruction in an order to get the solution.

In programming we take input from the user. The input in code formate. The input is given in compiler. The compiler converts the input into computer understandable language. Then the computer understands the task and executes it.

Reference:- https://www.coursera.org/in/articles/what-is-programming

Q3 What are the key steps involved in the programming process? Types of Programming Languages

the key steps in the programming process

Defining the problem
Planning and designing
Selecting a programming language
Writing code
Debugging and testing
Documenting
Optimizing

<u>Types of programming languages</u>

Functional programming languages
Procedural programming languages
Object-oriented programming languages (OOP)
Scripting languages
Logic programming languages

Q4 What are the main differences between high-level and low-level
Programming languages? World Wide Web & How Internet Works

High level languages

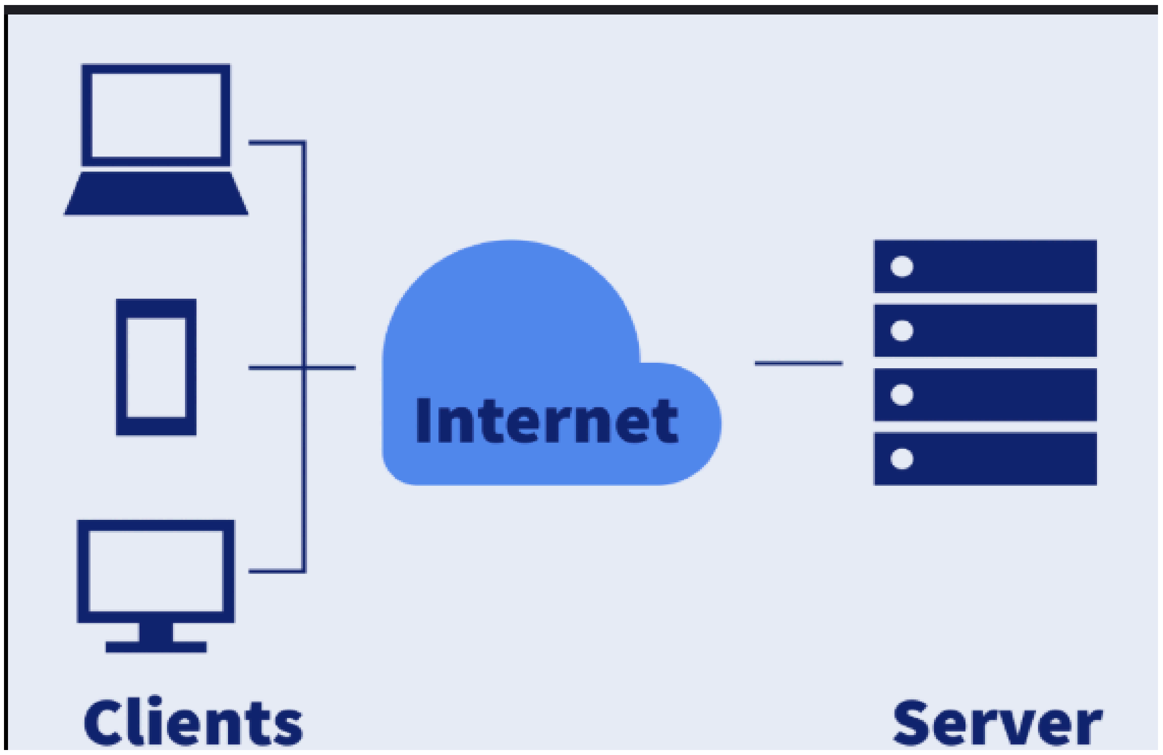
These languages are closer to human language, making them easier to learn, read, and write. High-level languages are designed to be human-friendly, and used to build websites and software. Examples of high-level languages include Python and Java.

Low-level languages

These languages are closer to machine code, giving programmers more direct control over the computer's hardware. They are more machine-friendly. They are harder for humans to understand, and require detailed knowledge of computer architecture.

Examples of low-level languages include Machine language and Assembly language

Q5 : Research and create a diagram of how data is transmitted from a client to a server over the internet.



Q6 Describe the roles of the client and server in web communication.

Client Sends requests to the server, specifying what information is needed.
Servers Listens for incoming requests from clients. Server Manages data storage and access control.

Q7 Network Layers on Client and Server

Physical Layer
Data Link Layer
Network Layer
Transport Layer
Session Layer
Presentation Layer
Application Layer

Q8 Explain Client Server Communication.

A method of communication where a client sends a request to a server, and the server responds.

Q9 Types of Internet Connections.

Broadband Connection.
Dial Up Connection.
Digital Subscriber Line (DSL)
Cable
Wi-Fi
Integrated Services Digital Network(ISDN)
Cellular Technology.

Q10 Research different types of internet connections (e.g., broadband, fiber, satellite)and list their pros and cons.

Broadband
Pros
Cheap
Data Caps
Speeds
Consistent

Cons
Fixed
Line Rental
Switching issues
Competition

Pros of Fiber
Connection Quality
Scalability
Security
Long-Term Cost-Effectiveness

Cons of Fiber Optics
Physical Damage
Short-Term Cost Effectiveness
Fiber Fuse
Unidirectional Light Propagation

Satellites pros
Global coverage
Flexibility
Solar power

Satellites cons
Signal interference
Higher costs
Technical issues
Delays
Single point of failure

Q11 How does broadband differ from fiber-optic internet?

Broadband is a general term for high-speed internet. In fiber optic internet is a specific type of broadband that uses thin glass or plastic fibers to transmit data, offering faster speeds and lower latency.

Q12 Simulate HTTP and FTP requests using command line tools (e.g., curl)

curl http://www.example.com - Sends a simple GET request to the specified URL.

curl -X POST -H "Content-Type: application/json" -d '{"key": "value"}'
https://api.example.com/endpoint - Sends a POST request with JSON data and a custom header

curl -T "local_file.txt" -u username:password ftp://ftp.example.com/remote_file.txt - Uploads a local file "local_file.txt" to a remote FTP server as "remote_file.txt" with username and password authentication.

Q13 What are the differences between HTTP and HTTPS protocols?

Hypertext transfer protocol (HTTP) is a protocol or set of communication rules for client-server communication. When you visit a website, your browser sends a HTTP request to the web server, which responds with an HTTP response. The web server and your browser exchange data as plaintext. In short, HTTP protocol is the underlying technology that powers network communication. As the name suggests, hypertext transfer protocol secure (HTTPS) is a more secure version or an extension of HTTP. In HTTPS, the browser and server establish a secure, encrypted connection before transferring data.

Q14 What is the role of encryption in securing applications?

Encryption plays a crucial role in securing applications by transforming sensitive data into an unreadable format

Encryption keeps apps safe by changing sensitive data into a code. Even if someone gets the data, they can’t read it without a key. It protects things like passwords, bank details, and personal info during use or storage.

Q15 Software Applications and Its Type

Software applications are programs made to do specific tasks. They help us with daily work, studies, fun, or business.

System Software
Helps run the computer.
Example: Windows, macOS, Linux.

Application Software
Used for specific tasks like typing or editing.
Example: MS Word, Excel, photo editing apps.

Utility Software
Keeps your computer running smoothly.
Example: Antivirus, file cleanup apps.

Web Applications
Works on the internet. No need to install.
Example: Gmail, YouTube, online shopping apps.

Mobile Applications (Apps)
Made for phones and tablets.
Example: WhatsApp, Instagram, Paytm.

Enterprise Software
Used by companies to manage work.
Example: Tally, business apps.

Gaming Software
Used to play games.
Example: PUBG, Ludo King, FIFA.

Multimedia Software
Used to create or watch videos, photos, or music.
Example: VLC Player, Adobe Photoshop.

Education Software
Used for learning or teaching.

Example: BYJU’S, Khan Academy, Udemy.

Database Software
Helps store and manage data.
Example: Microsoft Access, MySQL.

Q16 Identify and classify 5 applications you use daily as either system software
Or application software.

Android/Windows - System Software
Chrome- Application Software
Google Docs - Application Software
WhatsApp - Application Software
Telegram - Application Software

Q17 What is the difference between system software and application software?

System Software
It helps the computer to work.It controls the computer's parts and allows other software to run.It runs in the background. we don't use it directly.

Application Software
It helps you do specific tasks on the computer.It is for users to do things like typing, editing photos, or browsing. We open and use it directly.

Q18 Layers in Software Architecture.

Software architecture is how we organize an application

Presentation Layer
Application Layer
Business Logic Layer
Data Access Layer
Database Layer

Q19 source code

The term source code refers to the set of instructions and statements written by a programmer in a programming language. Source code is the foundation for any application, system, or program. It can be modified, shared, or maintained to update or improve functionality.

Q20 difference between source code and machine code.

Source code
It is written by people in programming languages like Python or C++.
It is easy for humans to read and understand.
Computers cannot understand it.

Machine code

It is written in 0s and 1s (binary).
Computers can understand and run it directly.
It is hard for humans to read.

Q21 Why is version control important in software development?

Collaboration
Change Tracking
Code Integrity
Accountability
Disaster Recovery
Efficiency in Development
Improved Deployment

Q22 types of software

System Software
Application Software
Utility Software
Middleware
Programming Software
Driver Software
Open-Source Software
Proprietary Software

Q23 Create a list of software you use regularly and classify them into the followingcategories: system, application, and utility software.

Windows - Operating System
Google Chrome - Web Browser
Antivirus Software - McAfee

Q24 What are the differences between open-source and proprietary software?

Open source
Open-source software provides free access to its source code, allowing users to view, modify, and distribute it.
Open-source software is usually free or available at a minimal cost.
Open-source software allows for extensive customization by users.
Open-source software is often community-owned or managed by organizations and shared under permissive licenses.
Open-source software relies on community-driven support, which may not always be consistent.
Open-source software may have vulnerabilities due to open access but benefits from community reviews and patches.
Open-source software examples include Linux, Mozilla Firefox, and LibreOffice.

Proprietary software
Proprietary software keeps its source code closed, preventing users from accessing or modifying it.
Proprietary software typically requires payment through a purchase or subscription.
Proprietary software offers limited or no customization options.
Proprietary software is owned by a company that restricts usage rights.
Proprietary software provides dedicated customer support services.
Proprietary software's security is managed by the vendor, requiring users to trust their policies.
Proprietary software examples include Microsoft Windows, Adobe Photoshop, and Microsoft Office.

Q25 Application Software

Application software is a type of program that helps you do specific tasks on a computer or phone. It helps you with things like writing, making presentations, or playing music.

Q26 Sdlc

The software development process is a series of steps to create a software program. It helps make sure the software is useful, works well, and meets the user's needs.

Here are the main steps in the process:

Planning: This is where the idea for the software is created. Developers decide what the software will do and how it will help users.

Design: In this step, the look and structure of the software are planned. It includes how it will work and how users will interact with it.

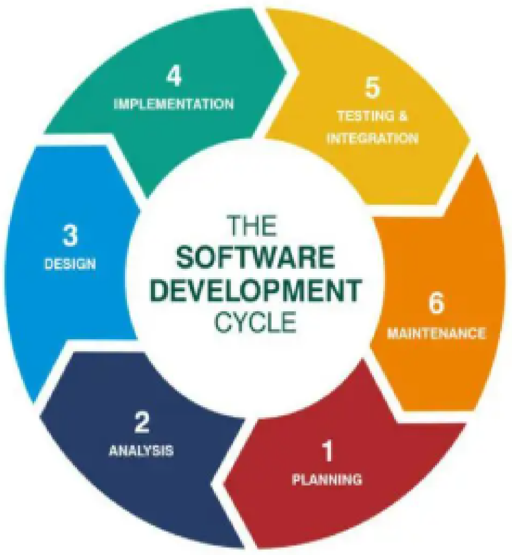
Development: This is when the coding happens. Developers write the code that makes the software work.

Testing: After development, the software is tested to find any problems or bugs. This helps make sure it works correctly.

Deployment: Once the software is tested and ready, it is released for users to download or use.

Maintenance: After the software is used, it may need updates or fixes. Developers continue to improve it and fix any new issues that arise.

Q27 Create a flowchart representing the Software Development Life Cycle (SDLC).



Q28 What are the main stages of the software development process?

Planning and Requirements:

First, understand what the software needs to do.
Talk to users to know their needs.
Plan the project with timelines and costs.

Design:

Plan how the software will work.
Create designs for the system and each part of it.
Sometimes, a prototype (a simple version) is made to show key features.

Development (Coding):

Write the code to make the software work.
Follow rules for clean and clear code.
Developers also test small parts of the software as they work.

Testing:

Check if the software works correctly.
Test its functions, speed, and security.
There are many types of testing, like unit testing and user testing.

Deployment:

Install the software for users.
It can be released slowly or all at once, depending on the plan.

Maintenance:

Fix problems that come up after the software is used.
Make updates to improve the software.
Add new features if needed.

Q29 Software Analysis

Software analysis is the process of checking how well a software works. It helps make sure the software does what it is supposed to do. Here are some key points:

Requirements Analysis: Understand what users need from the software.

Code Analysis: Check the software’s code for quality and safety.

Design Analysis: Look at how the software is built to make sure it works well and can grow in the future.

Performance Analysis: Check how fast the software runs and if it uses too many resources.

Security Analysis: Find and fix any weaknesses in the software that could allow attacks.

Usability Analysis: See if the software is easy to use for the users.

Test Analysis: Make sure the software is tested properly to work correctly in all cases.

Q30 What is the role of software analysis in the development process?

Software analysis plays an important role in the development process. It helps ensure the software is built the right way from the start.

Understanding Requirements: Software analysis helps gather and understand what users need. This makes sure the final product meets their expectations.

Identifying Problems Early: By checking the design, code, and performance early, analysis helps find problems before they become big issues.

Improving Quality: Regular analysis helps improve the software's quality by ensuring it is safe, reliable, and easy to use.

Making the Software Efficient: It helps check how well the software works, finding ways to make it faster and use fewer resources.

Saving Time and Money: By finding problems early and fixing them, software analysis helps avoid costly mistakes and delays in the future.

Ensuring Security: It helps find security risks and fixes them, making the software safer for users.

Q31 System Design

System design is the process of planning how a software system will work.

Requirement Gathering: Understand what users need.
System Architecture: Plan the overall structure and how parts will work together.
High-Level Design: Define the main components and their interactions.
Low-Level Design: Detail each component's functions.
Database Design: Plan how data will be stored and managed.
User Interface Design: Plan how users will interact with the system.
Security Design: Ensure the system is secure.

Q32 What are the key elements of system design?
The key elements of system design are:

Requirements: Understanding the user needs and goals of the system.

System Architecture: Defining the structure of the system, including major components and their interactions.

Components: Designing individual parts or modules of the system, each with specific functions.

Data Flow: Planning how data will move through the system and between components.

Database Design: Structuring how data will be stored, retrieved, and managed.

User Interface: Designing how users will interact with the system, focusing on ease of use.

Security: Ensuring the system is protected from threats and data is secure.

Scalability: Planning for the system to handle growth in users, data, and workload.

Performance: Ensuring the system works efficiently and meets speed and resource use requirements.

Maintainability: Designing the system in a way that it can be easily updated and fixed in the future.

Q34 Why is software testing important?

Software testing is the process of checking a software to find and fix mistakes or problems. It helps make sure the software works as expected and meets the needs of users. Testing is done to find errors, improve quality, and ensure the software is secure and easy to use.

Quality: It helps fix mistakes and make the software work well.
User Experience: It makes the software easy to use.
Security: It finds and fixes security problems.
Cost: It saves money by fixing issues early.
Rules: It ensures the software follows needed rules.
Trust: It builds trust in the software.
Speed: It checks if the software runs fast and smooth.

Q35 What types ofsoftware maintenance are there?

There are four main types of software maintenance:

Corrective Maintenance: Fixing errors or bugs in the software that are found after it is released.

Adaptive Maintenance: Updating the software to work with new hardware, software, or environments.

Perfective Maintenance: Improving the software by adding new features or making it more efficient.

Preventive Maintenance: Making changes to the software to avoid future problems or issues.

Q36 What are the key differences between web and desktop applications?

Web App: A web application runs in a web browser and is accessible from any device with an internet connection. It doesn't require installation, as it is hosted online and updates automatically. Data is stored on the cloud, and it depends on the internet and browser performance to function.

Desktop App: A desktop application runs directly on a computer's operating system and needs to be installed on each device. It works offline and stores data locally. It uses the computer's resources, which can lead to faster performance, but requires manual updates.

Q37 What role does UI/UX design play in application development?

UI (User Interface) design is about creating the layout, visual elements, and overall look of the app. It makes the app easy to navigate and visually appealing.

UX (User Experience) design focuses on the overall experience, making sure the app is easy to use, functional, and meets users' needs. It involves understanding user behavior, ensuring smooth navigation, and improving the app’s flow.

Q38 What are the differences between native and hybrid mobile apps?

Native Apps:
Native apps are made for one platform, like iPhone or Android, using special programming languages. They work faster and can use all phone features, but take more time and money to build and update.

Hybrid Apps:
Hybrid apps are made using web tools and work on both iPhone and Android. They are quicker and cheaper to build but may be slower and have limited access to phone features. Updates are easier because only one app needs to be updated.

Q39 What is the significance of DFDs in system analysis?

DFDs help understand, analyze, and improve systems.

Data Flow Diagrams (DFDs) are important because they:

Visualize System Processes:
Show how data flows through the system.

Simplify Complex Systems:
Break down systems into easy-to-understand parts.

Identify Issues:
Help find problems and gaps in the system.

Improve Communication:
Make it easier for team members to collaborate.

Aid Design and Documentation:
Serve as a blueprint for system development and future updates.

Q40 What are the pros and cons of desktop applications compared to webapplications?

Desktop Applications:
Pros:
Desktop apps typically offer better performance as they run directly on the device without relying on an internet connection. They can be used offline and provide more control over features and settings. They also have full access to device hardware, such as the camera and file system.

Cons:
The main downside is that desktop apps are limited to the device where they are installed. They require installation, taking up storage space, and updates need to be done manually. Additionally, desktop apps need separate versions for different operating systems, like Windows or macOS.

Web Applications:
Pros:
Web apps can be accessed from any device with an internet connection, making them more flexible and convenient. They don't require installation, as users can simply open them in a browser. Updates are done automatically, and web apps work across different platforms without needing separate versions.

Cons:
Web apps depend on a stable internet connection to function, which can limit their use offline. They may perform slower than desktop apps, especially for complex tasks. They also have limited access to device features, and data stored online could be more vulnerable to security issues.

Q41 How do flowcharts help in programming and system design?

flowcharts make programming and system design clearer and more efficient.Flowcharts help in programming and system design by:

Visualizing Logic: They show the steps and flow of a program clearly.
Planning and Designing: Help plan the structure and identify issues early.
Simplifying Complex Processes: Break down complex tasks into simple steps.
Communication: Make it easier to explain processes to both technical and non-technical people.
Debugging and Maintenance: Help locate errors and simplify updates or maintenance