Module 1 – Overview of IT Industry: Theory Answers

What is a Program?

A program is a set of instructions written in a programming language to perform a specific task. It functions by executing these instructions on a computer to manipulate data, make decisions, and produce output.

What is Programming?

Programming is the process of creating software by writing code in programming languages. It involves designing logic, writing instructions, testing, debugging, and maintaining the code.

Key Steps in Programming Process

1. Problem Understanding
2. Requirement Analysis
3. Algorithm Design
4. Coding
5. Testing & Debugging
6. Deployment
7. Maintenance

High-level vs Low-level Languages

High-level languages (e.g., Python, Java) are human-readable and abstract away hardware details. Low-level languages (e.g., Assembly) are closer to machine code and give more control over hardware.

Client & Server in Web Communication

Client initiates requests for services and resources. Server responds by processing the request and returning the result. For example, when accessing a website, the browser is the client, and the web server hosts and delivers the requested pages.

TCP/IP Model and its Layers

The TCP/IP model has four layers: Application, Transport, Internet, and Network Access. It governs how data is packetized, addressed, transmitted, routed, and received on the internet.

Client Server Communication

Client-server communication involves the client sending a request to the server, and the server processing that request and responding. This happens using network protocols like HTTP.

Broadband vs Fiber-optic Internet

Broadband uses copper wires and is slower with more signal degradation. Fiber-optic uses light signals through fiber cables, offering higher speed and reliability.

HTTP vs HTTPS

HTTPS is the secure version of HTTP. It uses SSL/TLS encryption to secure data transmission between client and server, preventing eavesdropping and tampering.

Role of Encryption

Encryption secures applications by encoding data so that only authorized parties can access it, protecting sensitive information during storage and transmission.

System vs Application Software

System software manages hardware (e.g., OS), while application software helps users perform tasks (e.g., Word, Excel).

Significance of Modularity in Software Architecture

Modularity allows dividing software into independent, manageable units. It improves maintainability, scalability, and team collaboration.

Importance of Layers in Software Architecture

Layers separate concerns like UI, business logic, and data. It improves code organization, reusability, and ease of maintenance.

Importance of Development Environment

A development environment provides tools and configurations for writing, testing, and debugging code efficiently.

Source Code vs Machine Code

Source code is human-readable; machine code is binary code that computers execute. Source code must be compiled or interpreted into machine code.

Importance of Version Control

Version control like Git tracks code changes, allows collaboration, rollback, and efficient project management.

Benefits of Using GitHub for Students

GitHub allows students to collaborate, showcase projects, track changes, and contribute to open-source, enhancing learning and visibility.

Open-source vs Proprietary Software

Open-source software's source code is freely available for use and modification. Proprietary software is owned and licensed, restricting access and modification.

GIT and Team Collaboration

GIT enables collaboration through features like branching, merging, and commit history. Teams can work simultaneously without conflict.

Role of Application Software in Businesses

Application software automates tasks like accounting, data analysis, and communication, improving productivity and decision-making.

Stages of Software Development Process

1. Requirement Gathering
2. System Design
3. Implementation
4. Testing
5. Deployment
6. Maintenance

Importance of Requirement Analysis

Requirement analysis ensures developers understand user needs, preventing costly errors and ensuring relevant functionality.

Role of Software Analysis

Software analysis defines what the system should do and identifies constraints, guiding design and development stages.

Key Elements of System Design

Includes architecture, user interfaces, database design, data flow, and system integration specifications.

Importance of Software Testing

Software testing ensures the system is bug-free, meets requirements, and provides a quality product to users.

Types of Software Maintenance

1. Corrective – fix bugs
2. Adaptive – adjust to environment
3. Perfective – enhance features
4. Preventive – avoid future issues

Web vs Desktop Applications

Web apps run in browsers, are platform-independent and easily updated. Desktop apps run locally, offer better performance but require installation.

Advantages of Web Applications

Accessible from anywhere, no installation, easy updates, and cost-effective deployment.

UI/UX Role in Development

Good UI/UX improves usability, engagement, and satisfaction, directly affecting product success.

Native vs Hybrid Mobile Apps

Native apps are platform-specific and faster. Hybrid apps use web technologies and work across platforms with one codebase.

Significance of DFDs

DFDs visually represent data flow and processes in a system, aiding analysis and communication with stakeholders.

Pros and Cons of Desktop Applications

Pros: Fast, full hardware access. Cons: Platform-dependent, need manual updates. Web apps are more accessible but slower.

How Flowcharts Help

Flowcharts visualize program logic, aiding planning, debugging, and understanding system processes.