Theory Exercise

A-1: A program is a **list of instructions** written in a computer language that tells the computer exactly what to do. It works by following those instructions step by step to complete a task.

A-2: The programming process involves several key steps to create a working program.

1. Requirement Analysis
2. Development
3. Testing
4. Fixing any errors
5. Maintaining

A-3:

High Level Languages: High-level languages are easy to read and write for humans. For example: Python, Java, C++, JavaScript

Low Level Languages

Low-level languages are the kind of languages which computers easily understand. For example: Assembly, Machine Code

A-4:

The **client** sends requests to the server when they want to access something on the internet.

The **server** gets the requests from clients and responds by sending the requested information back.

For example, when a client types a website address or clicks a link, the client's browser asks the server to send the information. And when the server receives a request for a webpage, it finds the page and sends it to the client's browser to display.

A-5:

The **TCP/IP model** is a set of rules that helps computers communicate over the internet or other networks. It breaks down communication into smaller parts called **layers**, each with a specific job.

There are **four main layers** in the TCP/IP model:

1. **Application Layer**
   Communication starts with this layer. It includes programs like web browsers and email apps that send and receive data. It makes sure data is in the right format for users.

2. **Transport Layer**
   This layer makes sure data is sent correctly between devices. It breaks data into smaller pieces and checks if everything arrives without errors. TCP (Transmission Control Protocol) works here.

3. **Internet Layer**
   This layer is responsible to send data across different networks. It uses IP (Internet Protocol) to address and route data packets.

4. **Network Access Layer** (also called Link Layer)
   This layer deals with the physical connection to the network, like cables and Wi-Fi. It sends data over the actual hardware.

A-6:

Client-server communication is **how a server responds to the client** and communicates based on the client's request.
The **client** is the device that wants some information or service, and it sends a request to the server.
The **server** is a powerful computer that waits for these requests and responds by sending the requested data or performing the service.

A-7 :
Fibre-Optics: JioFiber, Airtel Xstream Fiber, BSNL Fiber are examples of fibre optics. They are transmitting data using light signals

Broadband: Internet transmit using light connection to send data.

A-8 :
HTTP: HyperText Transfer Protocol [HTTP] is **not secure** and is represented as http://

HTTPS: HyperText Transfer Protocol Secure [HTTPS] is **secure** and is represented as https://

A-9 :
Encryption protects data by changing it into a **secret code** so that only authorized users can read it. It keeps sensitive information (like passwords, messages or payment details) safe from hackers.
Even if someone steals the data, they can’t understand it without the correct decryption key.

A-10 :
System software runs the computer. It manages hardware and basic system functions. Example: Windows, macOS, Linux (Operating Systems)

Application software helps users do specific tasks on the computer. Example: MS Word, Google Chrome, Photoshop

A-11 :

Modularity means **breaking a program into smaller,** separate parts called modules. Each module does a specific job and can work independently or with others.
This makes software easier to build, test, fix, and update. It also helps teams work on different parts of the program at the same time without interfering with each other.

A-12 :

Layers help organize software into different levels, where each layer has a specific job.
Each layer can be changed or updated without affecting the others, which improves flexibility and reduces errors. It also helps divide work among developers and keeps the code clean and structured.

A-13 :

A development environment is the setup (tools, software, and settings) used to write, test, and build software. It helps developers write code efficiently, find and fix errors, and see how the program works before releasing it.

A-14 :

Source Code is the code written by a programmer in a human-readable language like Python, C++, or Java.

Machine Code is the code that the computer actually understands — made of 0s and 1s (binary). It is created by converting source code using a compiler or interpreter.

A-15 :

Version control keeps track of every change made to the code during development.
It helps developers work together, go back to previous versions if something breaks, and avoid losing work. It also makes it easier to manage updates, fix bugs, and track who made which changes and why.

A-16 :

Github is very useful for students as they manage the following things easily:

1. Store and manage code online
2. Track changes
3. Collaborate on group projects
4. Build a portfolio to show their skills

A-17 :

Open-Source Software: The source code is freely available. Anyone can view, use, modify, and share it.
Example: Linux, VLC Media Player

Proprietary Software: The source code is private. Only the owner or company can modify or share it.
Example: Microsoft Windows, Adobe Photoshop

A-18 :

The GIT helps teams to work together in monitoring the change in code, so everyone knows who did what and when. A member of each team can work on its share without messing up the work of others. The GIT code is allowed to merge, solve conflicts and create mistakes easily, making the teamwork fast, safe and more organized.

A-19 :

The application helps software businesses to do specific functions such as finance, communication or making documents. It improves productivity by automating the work, organizing data and making processes rapidly and easier. Examples include accounting software, email customer and customer management systems.

A-20 :

1. Requirement Gathering : This is the first where developers meet with clients or users to understand and collect all the necessary details, features, and expectations.

2. Analysis: The team analyzes what is possible, checks technical needs, and identifies any risks or limitations.

3. Design : The team starts designing the structure, layout (UI), and how different parts of the software will interact. They create diagrams or wireframes to visualize the plan before coding.

4. Coding (Implementation) : This is where developers actually write the program using programming languages like Python, Java, or C++. They turn the design into a working product by building the features and functions.

5. Testing : After coding, the software is tested to find and fix any bugs or issues. Testing checks if the software is secure, and performs well as expected.

6. Maintenance : Even after deployment, work continues. Developers fix bugs, make updates, improve performance, or add new features. This ensures the software stays useful and up to date over time.

A-21 :

The **requirement analysis phase** is important because it helps the team clearly understand **what the software should do**. It ensures that the developer and client are on the same page about goals, features, and expectations.

Good analysis helps avoid confusion, saves time and money, and reduces the risk of errors later in the project. Without it, the software might not meet the user's needs or work correctly.

A-22 :

Software analysis helps in **understanding and organizing** what the software needs to do. It turns user requirements into clear, detailed information that developers can use to design and build the system.

This phase identifies **what is needed**, checks if it’s **possible**, and helps plan for problems early. It acts as a **bridge between planning and designing**, ensuring the software is useful, correct, and successful.

A-23 :

1. **Architecture Design** : Defines the overall structure of the system and how different parts interact.

2. **User Interface Design (UI)** : Focuses on how users will interact with the system (screens, buttons, layout).

3. **Database Design** : Plans how data will be stored, organized, and accessed efficiently.

4. **Implementation Design** – Makes sure the system runs smoothly, loads quickly, and handles users properly.

A-24 :

Software testing is important because it helps find and fix **errors or bugs** before the software is released.

It ensures the software **works correctly**, is **secure**, and meets the user's needs. Testing also improves **performance**, increases **user trust**, and helps avoid costly problems later. Without testing, software may crash, give wrong results, or cause serious issues for users.

A-25 :

There are 3 types of software maintenance which are as follows:

Corrective Maintenance : Fixes bugs and errors found after the software is in use.
Adaptive Maintenance : Updates the software to keep it compatible with new systems, tools, or environments.
Perfective Maintenance : Improves the software by adding new features or enhancing performance based on user needs.

A-26 :

Web App:

1. No installation needed, runs in a web browser.
2. Needs internet to run (mostly).
3. Can be accessed from any device with a browser.
4. Data is usually stored online (cloud).
5. Examples: Gmail, Google Docs, Canva

Desktop App:

1. Must be downloaded and installed on the computer.
2. Can work without the internet (in many cases).
3. Only works on the device where it is installed.
4. Data is stored on the local device.
5. Examples: MS Word, Photoshop, VLC Media Player

A-27 :

Advantages of using web applications are :

1. No Installation Needed
2. Access from Anywhere
3. Easy Updates
4. Platform Independent
5. Cost-Effective
6. Easy Collaboration
7. Centralized Data Storage

A-28 :

UI (User Interface) design focuses on how the app looks — the layout, colors, buttons, and screens.
UX (User Experience) design focuses on how the app feels — making sure it's easy, smooth, and enjoyable to use.

Good UI/UX design helps users **understand and use the app easily**, making them more likely to use it again. It also reduces confusion and errors, improves user satisfaction, and helps the app stand out in the market.

A-29 :

Native Mobile Apps:

1. Developed specifically for one platform (like Android or iOS).
2. Uses platform-specific languages (like Java/Kotlin for Android, Swift for iOS).
3. Offers faster performance and smooth user experience.
4. Has full access to device features like camera, GPS, and notifications.
5. Requires separate development for each platform, which increases time and cost.

Hybrid Mobile Apps:

1. Developed using one codebase for multiple platforms.
2. Uses web technologies like HTML, CSS, and JavaScript inside a native container.
3. Performance is slightly lower compared to native apps.
4. Access to device features is limited or depends on plugins.
5. Faster to develop and more cost-effective for cross-platform needs.

A-30 :

Data Flow Diagrams (DFDs) are a crucial tool in system analysis because they provide a visual representation of how data moves through a system, helping to understand the system's functionality and identify potential issues.

They make it easy to visualize the flow of data, making complex systems easier to grasp. DFDs can be used to analyze existing systems, design new ones, and align stakeholders around data handling.

A-31 :

Desktop Applications
Pros:

1. Work without an internet connection
2. Usually faster and more powerful
3. Can access all system resources and hardware
4. Better performance for heavy tasks like video editing or gaming

Cons:

1. Need to be installed and updated manually
2. Only available on the device where installed
3. Harder to share or collaborate in real time

4. May not work across different operating systems

Web Applications
Pros:

1. No installation needed, runs in a browser
2. Easy to update and maintain
3. Can be accessed from any device with internet
4. Great for collaboration and real-time sharing

Cons:

1. Needs internet connection to work
2. Slower performance for heavy tasks
3. Limited access to system resources
4. May have security and browser compatibility issues

A-32 :

Flowcharts aid in programming and system design by visually representing the logic and flow of processes, making them easier to understand, communicate, and debug.

They help in breaking down complex tasks into simpler steps, identifying potential issues, and creating a blueprint for implementation.