1). What is a Program?

❖ THEORY EXERCISE: Explain in your own words what a program is and how it functions.

ANSWER:- A program is a set of instructions written in a specific programming language that tells a computer what to do. These instructions are designed to perform a particular task or solve a specific problem. The instructions in a program are written in a logical sequence so the computer can execute them step by step.

How It Functions:

1. Writing the Program:

A programmer writes the code using a programming language like C, Python, or Java.

2. Compilation/Interpretation:

The program is either compiled or interpreted to convert the high-level code into machine code (binary) that the computer can understand.

3. Execution:

The computer reads and executes each instruction in the program in the order they are written, unless told to jump, loop, or branch based on certain conditions.

4. Input and Output:

The program may take input from the user, process it, and then produce an output.

5. Storage and Memory Use:

During execution, the program uses memory (RAM) to temporarily store data and variables. It may also save data permanently in storage devices if needed.❖ LAB EXERCISE: Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.

ANSWER:-

Hello World in C Language

#include <stdio.h>

```
int main()
{
printf("Hello, World!\n"); // Print Hello World to the console
return 0;
}
```

Hello World in Java

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello, World!"); // Print Hello World to the console }
}
```

2). What is Programming?

❖ THEORY EXERCISE: What are the key steps involved in the programming process?

ANSWER:- Programming is the process of designing, writing, testing, and maintaining a set of instructions that a computer can follow to perform specific tasks. These instructions are written using a programming language such as C, Java, Python, etc. Programming allows humans to communicate with computers and automate tasks ranging from simple calculations to complex operations like controlling robots or managing data.

1. Understanding the Problem

 Analyze the problem statement or user requirement.

 Break the problem down into smaller, manageable parts.

2. Planning the Solution

 Choose the right programming language and tools.

 Design algorithms or flowcharts to solve the problem logically.

3. Writing the Code

 Translate the algorithm into a programming language.

 Follow syntax rules of the language being used.

4. Compiling and Debugging

 Compile the code to check for syntax errors.

 Fix errors (bugs) and recompile until the code is error-free.

5. Testing the Program

 Run the program with various inputs to check correctness.

 Verify if the program meets the intended functionality.

6. Documentation

 Write comments and notes to explain how the program works.

 Helps others (and yourself) understand the code in the future.

7. Maintenance

 Update the program as requirements change.

 Fix issues or improve performance over time.

3). Types of Programming Languages

❖ THEORY EXERCISE: What are the main differences between high-level and low-level programming languages?

ANSWER:-

Programming languages are broadly classified into two main types:

1. **High-Level Programming Languages**
2. **Low-Level Programming Languages**

Feature	High-Level Programming Languages	Low-Level Programming Languages
Abstraction Level	High (closer to human language)	Low (closer to machine hardware)
Examples	Python, Java, C++, JavaScript	Assembly Language, Machine Code
Ease of Use	Easier to write, read, and understand	Harder to write and understand
Portability	Platform-independent (runs	Platform-dependent (specific to hardware)

	on many systems)	
Execution Speed	Slower compared to low-level languages	Faster, directly executed by the CPU
Control Over Hardware	Limited control	Provides direct access to memory and hardware
Compilation/Translation	Needs a compiler or interpreter	May require assembler or run as machine code
Use Cases	Application development, web development	System programming, embedded systems

4). World Wide Web & How Internet Works

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

ANSWER:-

1. Client Request:

- The user types a website URL (e.g., www.example.com) in a browser.
- The browser sends a request to a DNS server to resolve the domain name into an IP address.

2. DNS Resolution:

- The DNS server responds with the IP address of the web server hosting the website.

3. Internet Routing:

- The client's device uses the IP address to send a HTTP/HTTPS request

through the internet.

○ The request passes through routers, ISPs (Internet Service Providers), and network switches.

4. Server Response:

○ The server receives the request, processes it, and sends back the requested web page or data to the client.

5. Data Delivery:

○ The data (HTML, images, scripts, etc.) is sent back as packets via the same network path.

○ The client's browser receives the packets and reconstructs the web page for the user to see.❖ THEORY EXERCISE: Describe the roles of the client and server in web communication.

ANSWER:-

Client Role

 A client is typically a user's device (like a computer, smartphone, or tablet) that initiates a request for information or services over the internet.

 It runs a web browser or application which sends an HTTP/HTTPS request to the server.

 The client waits for a response and then displays the content (e.g., a web page, image, or video).

Example:

Typing www.google.com in a browser sends a request to Google's server to load the homepage.

Server Role

 A server is a powerful computer or system that stores websites, applications, databases, and other resources.

 It listens for incoming client requests and responds with the appropriate data (like HTML files, images, or videos).

 Servers can handle many client requests at once.

Example:

Google’s server receives the request from the browser and sends back the search page.

Client	Server
Initiates the request	Responds to the request
Sends data request via browser	Processes and returns requested data
Displays data to the user	Hosts and manages content or services

5). Network Layers on Client and Server

❖ LAB EXERCISE: Design a simple HTTP client-server communication in any language.

ANSWER:-

HTTP Client (HTML + JavaScript)

```
<!DOCTYPE html>
<html>
<head>
<title>HTTP Client</title>
</head>
<body>
<h2>HTTP Client Example</h2>
<button onclick="fetchData()">Send Request</button>
<p id="response"></p>
<script>
function fetchData() {
fetch('http://localhost:3000')
.then(response => response.text())
```

```
.then(data => {
document.getElementById('response').innerText = data;
error;
})
});
.catch(error => { document.getElementById('response').innerText = 'Error: ' +
}
</script>
</body>
</html>
```

HTTP Server (Node.js with Express)

```
// server.js
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
res.send('Hello from the Server!');
app.listen(port, () => {
console.log(`Server running at http://localhost:${port}`);
});
});
```

❖ THEORY EXERCISE: Explain the function of the TCP/IP model and its layers.

ANSWER:-

The TCP/IP model (Transmission Control Protocol/Internet Protocol) is a conceptual framework used to understand how data is transmitted over the internet. It defines how devices communicate across networks using standardized protocols.

It consists of four layers, each with specific functions to ensure that data is packaged, transmitted, and received correctly.

Layer Description

1. Application

Layer

- Closest to the user. - Provides network services like HTTP,

FTP, DNS. - Responsible for initiating communication.

2. Transport

Layer

- Manages end-to-end communication between devices. -

Ensures reliable delivery using protocols like TCP or faster but

unreliable delivery using UDP.

3. Internet

Layer

- Responsible for logical addressing and routing.

- Uses IP

addresses to send data between networks. - Protocols: IP,

ICMP, ARP.

4. Network

Access Layer

- Also called Link or Data Link Layer. - Deals with physical

transmission of data over cables, Wi-Fi, etc. - Combines

hardware (MAC address) and drivers.

6). Client and Servers

❖ THEORY EXERCISE: Explain Client Server Communication

ANSWER:-

Client-Server Communication is a model where two types of devices

(or programs) interact over a network:

- The client sends requests for data or services.
- The server processes the request and sends back the appropriate

response.

This model is the foundation of how the internet and most modern networks work.

 Request Initiation:

The client (e.g., a web browser) sends a request to the server (e.g., a web server) for some service (like opening a webpage).

 Data Processing:

The server receives the request, processes it (e.g., fetches data from a database), and prepares a response.

 Response Delivery:

The server sends the result back to the client, which displays it to the user.

 When you visit www.google.com, your **browser (client)** sends an HTTP request to **Google's server**.

 The server processes the request and sends back the homepage.

 Your browser displays the page to you.

7). Types of Internet Connections

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

ANSWER:-

1. Broadband (DSL or Cable)

 Description: High-speed internet delivered via telephone lines (DSL) or coaxial cables (cable broadband).

Pros	Cons
Widely available	Speed can vary with distance (DSL)

Faster than dial-up	Slower than fiber
Always-on connection	Shared bandwidth (cable) may slow speeds during peak times

2. Fiber Optic Internet

 Description: Uses light signals through glass cables for ultra-fast internet.

Pros	Cons
Extremely fast and reliable	Limited availability in rural areas
Great for streaming/gaming	More expensive installation cost
Higher upload and download speeds	

3. Satellite Internet

 Description: Internet delivered via satellites orbiting the Earth.

Pros	Cons
Available in remote areas	High latency (delay)

No phone lines or cables needed	Weather can affect signal
Easy installation	Data caps and lower speeds

4. Mobile Data (4G/5G)

 Description: Internet provided through cellular networks to smartphones, dongles, or routers.

Pros	Cons
Portable and wireless	Signal strength varies by location
5G offers high speeds	Data limits and higher costs
Good for light browsing	Not ideal for heavy downloads

5. Dial-Up (Legacy)

 Description: Old form of internet that uses telephone lines.

Pros	Cons
Extremely cheap	Very slow speed
Available in most places	Cannot use phone while connected
Good for basic text browsing	Not suitable for modern applications

❖ THEORY EXERCISE: How does broadband differ from fiber-optic internet?

ANSWER:-

Broadband is a general term used for high-speed internet that is always on. It includes technologies like DSL, cable, and fiber, but in most cases, when people refer to "broadband," they mean DSL or cable internet.

Fiber-optic internet, on the other hand, is a specific type of broadband that uses light signals through glass or plastic fibers to transmit data at very high speeds.

Feature	Broadband (DSL/Cable)	Fiber-Optic Internet
Technology Used	Electrical signals via copper cables	Light signals via glass fiber cables
Speed	Moderate to high (up to 100 Mbps–1 Gbps)	Very high (can exceed 1 Gbps to 10+ Gbps)
Upload Speed	Typically much slower than download	Upload and download speeds are similar
Latency	Higher latency	Low latency – ideal for gaming and video
Reliability	Affected by weather and distance	Highly reliable and stable
Availability	Widely available, even in rural areas	Limited availability in some locations
Cost	More affordable and established	Higher cost, especially for installation

8). Protocols

❖ LAB EXERCISE: Simulate HTTP and FTP requests using command line tools (e.g., curl).

ANSWER:-

 Simulate an HTTP Request

curl http://example.com

 What It Does:

 Sends an HTTP GET request to example.com.

 Displays the raw HTML content of the webpage in the terminal.

 Additional Example:

curl -I http://example.com

I fetches only the HTTP headers.

 Simulate an FTP Request

Command to List Files on an FTP Server:

curl ftp://ftp.example.com/ --user username:password

 What It Does:

▪ Connects to an FTP server using the given credentials.

▪ Lists the files in the root directory. Command to Download a File from FTP:

curl -O ftp://ftp.example.com/file.txt --user username:password

 Downloads file.txt from the FTP server.

 Note: Many public FTP servers don't require login; in that case, use --user anonymous :

Protocol	Simulated Using	Description
HTTP	curl http://...	Used for web page requests

FTP	curl ftp://...	Used for file transfer

❖ THEORY EXERCISE: What are the differences between HTTP and HTTPS protocols?

ANSWER:-

HTTP (HyperText Transfer Protocol) and HTTPS (HyperText Transfer Protocol Secure) are both protocols used for transferring data between a client (e.g., web browser) and a server (e.g., web server). The main difference between them is security.

Feature	HTTP	HTTPS
Full Form	HyperText Transfer Protocol	HyperText Transfer Protocol Secure
Security	Not secure	Secure using SSL/TLS encryption
Data Transmission	Data is sent in plain text	Data is encrypted, protecting it from hackers
URL Prefix	http://	https://
Port Used	Port 80	Port 443
Digital Certificate	Not required	Requires an SSL/TLS certificate
Use Case	Basic websites where security	Banking, shopping, login systems, etc.

	is not critical	
Browser Indicator	No lock symbol in address bar	Shows a padlock 🔒 symbol in the browser

9). Application Security

❖ LAB EXERCISE: Identify and explain three common application security vulnerabilities. Suggest possible solutions.

ANSWER:-

1. SQL Injection

 Description:

An attacker manipulates input fields (like login forms) to inject malicious SQL code into a database query. This can lead to unauthorized data access, modification, or even deletion.

 Example:

SELECT * FROM users WHERE username = 'admin' --' AND password = 'password';

Solution:

 Use prepared statements and parameterized queries.

 Validate and sanitize all user inputs.

 Use ORM (Object Relational Mapping) libraries where possible.

2. Cross-Site Scripting (XSS)

 Description:

XSS allows attackers to inject malicious JavaScript into webpages viewed by other users, potentially stealing cookies or session

data.

 Example:

<script>alert('Hacked!');</script>Solution:

  Escape and sanitize user input before rendering on web

pages.

  Use Content Security Policy (CSP) headers.

  Use secure frameworks that automatically encode output

(e.g., React, Angular).

3. Insecure Authentication

 Description:

Weak login systems can allow attackers to guess or brute-force

passwords, or exploit sessions to gain unauthorized access.

 Solution:

 Enforce strong password policies.

 Implement multi-factor authentication (MFA).

 Use secure session management, such as expiring sessions

and using HTTPS.

Vulnerability Description Solution

SQL Injection Injecting malicious SQL

into queries

Use parameterized queries,

input validation

Cross-Site Scripting Injecting scripts into web

pages

Sanitize inputs, use CSP, secure

frameworks

Insecure

Authentication

Weak login and session

protection

Use MFA, strong passwords,

session security❖ THEORY EXERCISE: What is the role of encryption in securing applications?

ANSWER:-

Encryption is the process of converting readable data (plaintext) into an unreadable format (ciphertext) using a mathematical algorithm and a key. Only those with the correct decryption key can convert it back into its original form.

 Protects Data in Transit

 Encryption ensures that data sent between a client and a server (e.g., login details, personal information) cannot be read or altered by attackers while traveling across the internet.

 Example: HTTPS uses SSL/TLS to encrypt web traffic.

 Secures Stored Data (Data at Rest)

 Encryption is used to secure files, databases, and sensitive records stored on servers or devices.

 Even if hackers access the storage, encrypted data remains unreadable without the key.

 Ensures Confidentiality

 Prevents unauthorized users from accessing sensitive data, such as passwords, credit card numbers, or health records.

 Enhances User Trust and Compliance

 Users are more likely to trust applications that protect their data.

 Encryption helps applications comply with security regulations like **GDPR**, **HIPAA**, or **PCI-DSS**.

 Mitigates Impact of Data Breaches

 If encrypted data is stolen, it is useless without the decryption key, reducing the

risk of data leakage.

Type Purpose

Symmetric Encryption Same key for encryption and decryption (e.g., AES)

Asymmetric Encryption Public key to encrypt, private key to decrypt (e.g., RSA)

Hashing One-way encryption used for passwords (e.g., SHA-256)**10). Software Applications and Its Types**

❖ LAB EXERCISE: Identify and classify 5 applications you use daily as either system software or application software.

ANSWER:-

Software Name Description Type

Google Chrome Web browser used to access websites and web apps Application Software

Windows 10/11 Operating system that manages computer hardware System Software

Microsoft Word Word processor used to create and edit documents Application Software

Antivirus (e.g., Quick Heal) Protects the system from

malware and viruses

System

Software

Spotify Music streaming app for listening

to songs and podcasts

Application

Software

 System Software: Software that manages hardware and provides

a platform for running application software (e.g., operating

systems, device drivers, antivirus).

 Application Software: Software designed to perform specific tasks

for users (e.g., word processors, browsers, media players).❖ THEORY EXERCISE: What is the difference between system

software and application software?

ANSWER:-

Type of Software	Description
System Software	A type of software designed to run a computer's hardware and basic system operations. It acts as a platform for other software.
Application Software	A type of software designed to help the user perform specific tasks such as writing, browsing, or playing media.

Feature	System Software	Application Software
Purpose	Manages hardware and system operations	Helps users perform specific tasks
User Interaction	Works in the background	Directly interacted with by users
Examples	Operating systems (Windows, Linux), device drivers, antivirus	MS Word, Google Chrome, VLC Media Player
Dependency	Must be present for system operation	Depends on system software to run
Installation	Installed with or before application software	Installed by the user as needed

11). Software Architecture

❖ LAB EXERCISE: Design a basic three-tier software architecture diagram for a web application.

ANSWER:-

The Three-Tier Client-Server Architecture divides systems into presentation, application, and data layers, increasing scalability, maintainability, and efficiency. By separating the concerns, this model optimizes resource management and allows for independent scaling and updates, making it a popular choice for complex distributed systems.

❖ THEORY EXERCISE: What is the significance of modularity in software architecture?

ANSWER:-

Modularity in software architecture refers to the design principle of breaking a software system into separate, independent, and interchangeable components or modules, each responsible for a specific functionality.

Benefit	Explanation
Maintainability	Each module can be updated, fixed, or replaced without affecting the rest of the system.
Testability	Individual modules can be tested separately, making debugging easier.
Reusability	Modules can be reused across different projects or parts of a project.
Team Collaboration	Multiple developers can work on different modules simultaneously.
Scalability	Easier to scale the application by adding new modules or upgrading existing ones.
Separation of Concerns	Each module focuses on a single concern, improving clarity and structure.

Example

In a web application:

 Login module handles authentication.

 User module manages user profiles.

 Payment module handles transactions.**12). Layers in Software Architecture**

❖ LAB EXERCISE: Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.

ANSWER:-

Case Study: Online Bookstore System

Let's examine a simple Online Bookstore Web Application structured using three-layer architecture:

1 Presentation Layer (User Interface)

 Purpose: This layer is responsible for displaying information to users and collecting input.

 Technology Example: HTML, CSS, JavaScript, React

 Functionality:

- Displays book catalog
- Provides forms for user login, registration, and checkout
- Collects search input from the user
- Shows cart details and confirmation messages

2 Business Logic Layer (Application Layer)

 Purpose: Handles the core functionality and business rules of the application.

 Technology Example: Java, Python, C#, Node.js

 Functionality:

- Validates user login credentials
- Applies discount policies
- Calculates total price and taxes
- Controls access to features based on user role
- Manages order processing and inventory checks3 Data Access Layer (Database Layer)

 Purpose: Manages all interactions with the database.

 Technology Example: SQL, MySQL, PostgreSQL, MongoDB

 Functionality:

○ Retrieves books from the database

○ Saves user registration details

○ Updates stock levels after purchase

○ Fetches order history for each user

○ Handles secure storage of passwords

 A customer visits the bookstore website (Presentation Layer), searches for a book, and places an order. The Business Logic Layer processes the request, checks if the book is in stock, applies discounts, and sends the information to the Data Access Layer, which updates the database and returns confirmation.

The three-layer architecture ensures:

 Better organization of code

 Easier debugging and maintenance

 Scalability and separation of responsibilities❖ THEORY EXERCISE: Why are layers important in software

architecture?

ANSWER:-

Layers in software architecture represent logical divisions of responsibility within an application. Each layer handles specific tasks and communicates with the layers directly above or below it.

Benefit Description

Separation of

Concerns

Each layer focuses on a single responsibility (UI, logic,

data), making the system more organized.

Modularity Layers allow developers to modify or update one part of

the system without affecting others.

Easier Testing and

Debugging

Each layer can be tested independently for better fault

isolation.

Reusability Business logic or data access layers can be reused across

different projects or platforms.

Scalability New features can be added to specific layers without

redesigning the whole system.

Team Collaboration Teams can work in parallel on different layers (UI team,

backend team, database team).

Example

In a typical 3-layer architecture:

- The Presentation Layer handles user interaction.
- The Business Logic Layer processes data and applies rules.
- The Data Layer manages the database operations.

13). Software Environments

❖ LAB EXERCISE: Explore different types of software environments (development, testing, production). Set up a basic environment in a virtual machine

ANSWER:-

Objective

- Understand the purpose of different software environments.
- Set up a basic development environment using a virtual machine (VM).

Environment Purpose Characteristics

Development For writing and

debugging code

Includes tools like IDEs, compilers, and libraries.

Frequent changes happen here.

Testing To test code before

deployment

Simulates production-like conditions. Used for

unit, integration, and system testing.

Production Live environment for

end-users

Stable, optimized, and secure. Changes are

carefully deployed.

Step 1: Install a Virtual Machine Software

 Use VirtualBox or VMware Workstation Player (both free).

Step 2: Download and Install an OS

 Example: Ubuntu Linux ISO from ubuntu.com

Step 3: Create a New Virtual Machine

 Allocate CPU, RAM (at least 2GB), and disk space (at least 20GB).

 Attach Ubuntu ISO and install the OS.

Step 4: Install Development Tools

Once Ubuntu is installed:sudo apt update

sudo apt install build-essential git python3-pip

Install a code editor (like VS Code):

sudo snap install code –classic

Step 5: Set Up a Sample Project

Create a folder and basic "Hello World" Python file:

mkdir dev-project

cd dev-project

echo 'print("Hello, Developer!")' > hello.py

python3 hello.py

Summary

 Development Environment: Where developers build and test code.

 Testing Environment: Where QA teams verify software.

 Production Environment: Where real users access the application.❖ THEORY EXERCISE: Explain the importance of a development

environment in software production.

ANSWER:-

A development environment is a setup where software developers write, test, and debug code before it is moved to testing or production environments. It usually includes:

- Code editor or IDE (e.g., VS Code, Eclipse)
- Compiler/interpreter
- Libraries and frameworks
- Debugging tools
- Version control (e.g., Git)

Reason Explanation

Safe Testing Developers can experiment and test code without affecting live systems or users.

Tool Integration All necessary tools (IDEs, debuggers, compilers) are available in one place for efficient coding.

Code Isolation Bugs, crashes, or failures in the development environment do not impact other systems.

Faster Debugging Developers can quickly fix issues before passing the code to the testing or production stage.

Learning and Innovation

It provides a safe space for learning new technologies and trying innovative approaches.

Team Collaboration Multiple developers can work on features in parallel using shared development environments.

For instance, while building a web application, a developer uses a local development environment to:

- Write HTML, CSS, and JavaScript
- Test server-side code using a local database

 Run the application on localhost

 Fix bugs before deploying to the test server**14). Source Code**

❖ LAB EXERCISE: Write and upload your first source code file to Github.

ANSWER:-

Objective:

 Write a simple program (source code).

 Upload it to a GitHub repository.

Steps to Follow:

1. Create a Simple Program (e.g., in C or Python)

hello.py (Python Example):

print("Hello, GitHub!")

2. Create a GitHub Account

- Go to https://github.com
- Sign up (if you don't have an account)

3. Create a New Repository

- Click "New" under Repositories
- Name it hello-world
- Make it Public
- Add a README.md file (optional)

4. Upload the File

- Go to the repository
- Click "Add file" > "Upload files"
- Select your hello.py file and Commit changes❖ THEORY EXERCISE: What is the difference between source code and machine code?

ANSWER:-

Feature	**Source Code**	**Machine Code**
Definition	Human-readable set of programming instructions	Binary code understood by the computer (CPU)
Readability	Readable by programmers (e.g., Python, C)	Not human-readable (0s and 1s)
Conversion	Needs to be compiled/interpreted	Already executable by the machine
Modification	Can be edited and updated easily	Very difficult to understand or change
Storage	Stored in `.c`, `.py`, `.java` files etc.	Stored in `.exe`, `.bin`, or memory

Summary:

- Source Code: The original instructions written by a developer.
- Machine Code: The converted form that computers execute directly.
- GitHub is a platform to store, share, and manage source code with version control.**15). Github and Introductions**

❖ LAB EXERCISE: Create a Github repository and document how to commit and push code changes.

ANSWER:-

Objective:

Learn how to use GitHub to manage and push source code using version control.

Steps to Create a GitHub Repository and Push Code:

1. Create a GitHub Repository

 Go to https://github.com

 Sign in

 Click "New" (top-left under Repositories)

 Enter repository name (e.g., my-first-repo)

 Choose Public and click Create repository

2. Set Up Git Locally

If Git is not installed, download it from https://git-scm.com and install.

Open Command Prompt or Terminal and run:

```
git config --global user.name "Your Name"
```

```
git config --global user.email you@example.com
```

3. Initialize Your Project Folder

```
mkdir my-first-repo
```

```
cd my-first-repo
```

```
git init
```

4. Create a Sample File

```
echo "Hello GitHub!" > hello.txt
```5. Add and Commit the File

```
git add hello.txt
```

```
git commit -m "Initial commit"
```

6. Push to GitHub Repository

First, link your local folder to GitHub:

```
git remote add origin https://github.com/your-username/my-first-repo.git
```

Then push your code:

```
git push -u origin master
```❖ THEORY EXERCISE: Why is version control important in software

development?

ANSWER:-

Benefit Explanation

Tracks Changes Keeps a history of all modifications, allowing rollback to previous versions.

Supports Collaboration Multiple developers can work on the same project without overwriting each other's work.

Safe Experimentation You can create branches to test new features without affecting the main codebase.

Documentation Every change includes a message explaining why it was made (via commit messages).

Time Travel Revert to previous versions if something breaks or a bug is introduced.

Backup Ensures your code is safe in the cloud (GitHub, GitLab, Bitbucket, etc.).

16). Student Account in Github

❖ LAB EXERCISE: Create a student account on Github and collaborate on a small project with a classmate.

ANSWER:-

Objective:

To create a GitHub Student account and collaborate with a classmate on a shared

repository.

Steps to Create a Student GitHub Account and Collaborate:

1. Sign Up for GitHub

 Go to https://github.com/join

 Fill in your email, username, and password

 Click Create account

2. Apply for GitHub Student Developer Pack (Optional but Recommended)

 Go to https://education.github.com/pack

 Click “Get student benefits”

 Sign in and verify your student email address or upload a school ID

 Get free tools like Canva Pro, Replit Pro, JetBrains IDEs, and more

3. Create a Repository for the Project

 Click “New Repository”

 Name it something like student-collab-project

 Add a README.md file

 Set it to Public or Private

4. Invite a Collaborator

 Go to the repository

 Click on “Settings” > “Collaborators”

 Enter your classmate’s GitHub username and click “Add”

5. Work Together

 Both students can **clone**, **edit**, **commit**, and **push** code to the repository

 Use GitHub Issues or Pull Requests to collaborate more effectively❖ THEORY EXERCISE: What are the benefits of using Github

for students?

ANSWER:-

Benefit Explanation

Real-World

Experience

Students learn how professional developers

collaborate on real projects.

Collaboration Skills Encourages teamwork and code sharing with

classmates or global peers.

Access to

Developer Tools

GitHub Student Pack offers free premium tools like

Replit, JetBrains, etc.

Safe Backup Stores code securely online with version control.

Portfolio Building Students can showcase their projects to future

employers or universities.

Experiment

Without Risk

Use branching to try out new features or changes

without breaking the main project.

Track Progress Every code change is documented, helping track

learning and improvement.**17). Types of Software**

❖ LAB EXERCISE: Create a list of software you use regularly and

classify them into the following categories: system, application,

and utility software.

ANSWER:-

Software

Name

Category Purpose

Windows 11 System Software Operating system that manages hardware

and software resources.

Google Chrome Application

Software

Web browser used for accessing the

internet.

MS Word Application

Software

Used for word processing and creating

documents.

VLC Media

Player

Application

Software

Plays audio and video files.

WinRAR Utility Software Compresses and decompresses files.

Antivirus (e.g.,

Avast)

Utility Software Protects the computer from viruses and

malware.

File Explorer System Software Built-in file management tool in Windows

OS.

Calculator Application

Software

Performs mathematical calculations.❖ THEORY EXERCISE: What are the differences between open-source

and proprietary software?

ANSWER:-

Feature Open-Source

Software

Proprietary Software

Access to Source

Code

Yes, freely available to

view or modify

No, source code is closed

and protected

Cost Often free of charge Usually requires a paid

license or subscription

Customization Highly customizable by

users or developers

Customization is limited

or not allowed

Community

Support

Community-driven

development and support

Official support provided

by the company

Ownership Owned by the community

or non-profit

organizations

Owned and controlled by

a company or developer

Examples Linux, LibreOffice, Mozilla

Firefox

Windows OS, MS Office,

Adobe Photoshop**18). GIT and GITHUB Training**

❖ LAB EXERCISE: Follow a GIT tutorial to practice cloning, branching,

and merging repositories.

ANSWER:-

Objective: Practice essential Git operations used in team collaboration.

1. Clone a Repository:

git clone https://github.com/your-username/your-repo.git

2. Create and Switch to a New Branch:

git checkout -b feature-branch

3. Make Changes and Commit:

git add .

git commit -m "Added new feature or update"

4. Switch Back to Main Branch:

git checkout main

5. Merge the Branch into Main:

git merge feature-branch

6. Push Changes to GitHub:

git push origin main❖ THEORY EXERCISE: How does GIT improve collaboration in a

software development team?

ANSWER:-

Aspect Explanation

Version Control Git tracks all code changes, allowing developers to

revert to previous versions easily.

Branching Developers can work on separate features/bugs

independently without affecting the main code.

Collaboration Multiple team members can contribute to the same

project simultaneously.

Testing & Review Changes can be reviewed via pull requests before

being merged.

History &

Accountability

Every commit includes the author's name, message,

and timestamp.

Remote Access via

GitHub

Teams can access the codebase from anywhere,

increasing productivity.**19). Application Software**

❖ LAB EXERCISE: Write a report on the various types of application software and how they improve productivity.

ANSWER:-

Type	Examples	How They Improve Productivity
Word Processing Software	Microsoft Word, Google Docs	Helps create, format, and edit text documents quickly and efficiently.
Spreadsheet Software	Microsoft Excel, Google Sheets	Allows for complex data analysis, calculations, and visualizations.
Presentation	Microsoft	Helps in creating visually appealing

Software	PowerPoint, Canva	presentations for communication and training.
Database Management Software	MySQL, Microsoft Access	Stores and manages large volumes of structured data, improving data organization and retrieval.
Communication Software	Zoom, Microsoft Teams, Slack	Enhances team collaboration through messaging, video calls, and file sharing.
Graphic Design Software	Adobe Photoshop, CorelDRAW	Enables creation of visuals, improving branding and marketing materials.
Web Browsers	Chrome, Firefox, Safari	Provides access to online tools, platforms, and research for work and learning.
Accounting Software	Tally, QuickBooks	Automates financial tracking, invoicing, and reporting for businesses.