Assignment Module-1

1. What is a Program?

LAB EXERCISE:

Write a simple "Hello World" program in two different programming languages. Compare their structure and syntax.

Answer:

In C:

```
#include <stdio.h>
int main() {
printf("Hello, World!\n");
return 0;
}
```

In Python:

```
print("Hello, World!")
```

The C program requires including a header file, defining the main function, using curly braces to mark the function body, and ending each statement with a semicolon. Python uses a single line, no main function, and no semicolons. C is compiled and lower-level, while Python is interpreted and high-level[1][2].

THEORY EXERCISE:

Explain in your own words what a program is and how it functions.

Answer:

A program is a collection of instructions written in a programming language like C. These instructions tell the computer what tasks to perform, such as calculations or displaying information. The computer executes the program step by step, following the logic defined by the programmer. Programs can be simple, like printing text, or complex, like operating systems[1][3][4].

2. What is Programming?

THEORY EXERCISE:

What are the key steps involved in the programming process?

Answer:

Key steps include:

- **Understanding the problem**: Clearly define what needs to be solved.
- **Designing a solution**: Plan the logic and structure, often using flowcharts or pseudocode.

- **Writing the code**: Implement the solution in a programming language like C.
- **Compiling and debugging**: Convert code to machine language and fix errors.
- **Testing**: Ensure the program works as expected with different inputs.
- **Documentation and maintenance**: Write comments, user guides, and update the program as needed[3][4].

3. Types of Programming Languages

THEORY EXERCISE:

What are the main differences between high-level and low-level programming languages?

Answer:

- **High-level languages** (e.g., C, Python) are easier for humans to read and write, have more abstraction from hardware, and include features like variables, loops, and functions.
- **Low-level languages** (e.g., Assembly, machine code) are closer to hardware, offer more control, but are harder to read and write.
- High-level languages are portable across systems, while low-level languages are often hardware-specific[3][4].

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:

When a client (like a web browser) requests a webpage, the request travels through the local network, then through routers and switches to the Internet Service Provider (ISP), across the internet, and finally reaches the server. The server processes the request and sends data back along the same path.

Diagram:

Client → Router → ISP → Internet → Server → (response returns the same way).

THEORY EXERCISE:

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

Answer:

The client (e.g., browser) initiates requests for data or services, and the server responds by providing the

requested resources or performing actions. The client displays the results to the user, while the server handles processing and data storage.

5. Network Layers on Client and Server

LAB EXERCISE:

Design a simple HTTP client-server communication in C.

Answer:

A basic C client uses sockets to send an HTTP GET request to a server. The server listens on a port, accepts connections, reads the request, and sends back a response.

Example:

Client: uses `socket()`, `connect()`, `send()`, and `recv()` functions.

Server: uses `socket()`, `bind()`, `listen()`, `accept()`, `recv()`, and `send()`.

THEORY EXERCISE:

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

Answer:

The TCP/IP model has four layers:

- **Application**: Handles network processes (e.g., HTTP, FTP).
- **Transport**: Provides reliable data transfer (e.g., TCP, UDP).
- **Internet**: Routes data packets (e.g., IP).
- **Network Access**: Manages physical transmission (e.g., Ethernet).
  Each layer has a specific role in ensuring data is correctly sent and received across networks.

6. Client and Servers

THEORY EXERCISE:

Explain Client-Server Communication.

Answer:

In client-server communication, the client sends requests for resources or services, and the server processes these requests and sends responses. Communication happens over a network using protocols like HTTP or FTP, and the server can serve multiple clients simultaneously.

7. Types of Internet Connections

Assignment Module-1

LAB EXERCISE:

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

Answer:

- **Broadband**: Widely available, moderate speed, affordable, but speed drops with distance.
- **Fiber**: Very high speed, reliable, not affected by distance, but expensive and limited availability.
- **Satellite**: Available in remote areas, but high latency, affected by weather, and costly.

THEORY EXERCISE:

How does broadband differ from fiber-optic internet?

Answer:

Broadband uses copper or coaxial cables and offers moderate speeds, while fiber-optic uses glass fibers to transmit data as light, allowing for much faster and more reliable connections.

8. Protocols

LAB EXERCISE:

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

Answer:

To make an HTTP request: `curl http://example.com`

To make an FTP request: `curl ftp://ftp.example.com/file.txt`

These commands show how to interact with web and file transfer servers using different protocols.

THEORY EXERCISE:

What are the differences between HTTP and HTTPS protocols?

Answer:

HTTP transfers data in plain text and is not secure. HTTPS encrypts data using SSL/TLS, protecting it from interception and tampering.

9. Application Security

LAB EXERCISE:

Identify and explain three common application security vulnerabilities. Suggest solutions.

Answer:

Assignment Module-1

1. **SQL Injection**: Attacker injects malicious SQL code. Solution: Use parameterized queries.
2. **Cross-Site Scripting (XSS)**: Attacker injects scripts into web pages. Solution: Sanitize user input.
3. **Buffer Overflow**: Attacker overflows memory buffer. Solution: Validate input size and use safe functions.

THEORY EXERCISE:

What is the role of encryption in securing applications?

Answer:

Encryption converts data into unreadable code, protecting sensitive information during storage and transmission so only authorized parties can access it.

10. Software Applications and Its Types

LAB EXERCISE:

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

Answer:

- Windows OS (System software)
- Microsoft Word (Application software)
- Google Chrome (Application software)
- Antivirus (Utility software)
- File Explorer (System software)

THEORY EXERCISE:

What is the difference between system software and application software?

Answer:

System software manages hardware and provides a platform for applications (e.g., OS), while application software helps users perform specific tasks (e.g., word processing).

11. Software Architecture

LAB EXERCISE:

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

Assignment Module-1

Answer:

Three tiers:

- **Presentation Layer**: User interface
- **Business Logic Layer**: Application logic
- **Data Access Layer**: Database interaction
  Each layer is separated for better maintainability and scalability.

THEORY EXERCISE:

What is the significance of modularity in software architecture?

Answer:

Modularity divides software into independent components, making development, testing, and maintenance easier and allowing code reuse.

12. Layers in Software Architecture

LAB EXERCISE:

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

Answer:

In a library management system:

- Presentation layer displays book listings and accepts user input.
- Business logic layer processes book borrowing and returning.
- Data access layer manages storage and retrieval of book and user records.

THEORY EXERCISE:

Why are layers important in software architecture?

Answer:

Layers separate concerns, making systems easier to maintain, test, and scale, and allow teams to work on different parts independently.

13. Software Environments

Assignment Module-1

LAB EXERCISE:

Explore different software environments (development, testing, production). Set up a basic environment in a VM.

Answer:

- **Development**: Programmers write and test code.
- **Testing**: QA team tests features and finds bugs.
- **Production**: End-users access the final product.
  In a VM, install a C compiler and editor to create a development environment.

THEORY EXERCISE:

Explain the importance of a development environment in software production.

Answer:

A development environment provides tools and settings for safe, efficient coding and testing, preventing errors from affecting the live system.

14. Source Code

LAB EXERCISE:

Write and upload your first source code file to GitHub.

Answer:

Create a file called hello.c with:

#include <stdio.h>

int main() { printf("Hello, GitHub!\n"); return 0; }

Upload it to a new GitHub repository using the website or `git` commands.

THEORY EXERCISE:

What is the difference between source code and machine code?

Answer:

Source code is human-readable instructions, while machine code is binary instructions executed by the computer's processor.

15. GitHub and Introductions

Assignment Module-1

LAB EXERCISE:

Create a GitHub repository and document how to commit and push code changes.

Answer:

1. Create a repository on GitHub.
2. Clone it: `git clone <repo_url>`
3. Add files: `git add filename.c`
4. Commit: `git commit -m "Initial commit"`
5. Push: `git push`

THEORY EXERCISE:

Why is version control important in software development?

Answer:

Version control tracks changes, enables collaboration, allows reverting to previous versions, and helps manage multiple features or bug fixes.

16. Student Account in GitHub

LAB EXERCISE:

Create a student account on GitHub and collaborate on a small project with a classmate.

Answer:

Sign up with your student email, create a repository, invite your classmate as a collaborator, and both contribute code and track changes.

THEORY EXERCISE:

What are the benefits of using GitHub for students?

Answer:

GitHub helps students learn collaboration, version control, build portfolios, and access open-source code and resources.

17. Types of Software

LAB EXERCISE:

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

Answer:

- System: Linux OS
- Application: VS Code, Chrome
- Utility: Disk Cleanup, Antivirus

THEORY EXERCISE:

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

Answer:

Open-source software shares its source code and can be modified by anyone. Proprietary software is closed source and owned by companies.

18. Git and GitHub Training

LAB EXERCISE:

Follow a Git tutorial to practice cloning, branching, and merging repositories.

Answer:

Clone: `git clone <repo_url>`

Branch: `git checkout -b new-feature`

Merge: `git checkout main; git merge new-feature`

THEORY EXERCISE:

How does Git improve collaboration in a software development team?

Answer:

Git allows multiple developers to work on the same project, manage changes, resolve conflicts, and merge contributions efficiently.

19. Application Software

LAB EXERCISE:

Write a report on various types of application software and how they improve productivity.

Answer:

Word processors (e.g., MS Word), spreadsheets (Excel), browsers (Chrome), and email clients (Outlook) help users create documents, analyze data, browse the web, and communicate, increasing efficiency and productivity.

Assignment Module-1

THEORY EXERCISE:

What is the role of application software in businesses?

Answer:

Application software automates business tasks, manages data, improves communication, and streamlines operations, leading to better productivity.

20. Software Development Process

LAB EXERCISE:

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

Answer:

Typical SDLC flow:

Requirement Gathering → Analysis → Design → Implementation → Testing → Deployment → Maintenance

THEORY EXERCISE:

What are the main stages of the software development process?

Answer:

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

21. Software Requirement

LAB EXERCISE:

Write a requirement specification for a simple library management system.

Answer:

The system must allow users to search for books, borrow and return books, and manage user accounts. Admins can add or remove books and view borrowing history.

Assignment Module-1

THEORY EXERCISE:

Why is the requirement analysis phase critical in software development?

Answer:

It ensures developers understand user needs, reducing the risk of building the wrong product and saving time and resources.

22. Software Analysis

LAB EXERCISE:

Perform a functional analysis for an online shopping system.

Answer:

Key functions: user registration, product browsing, cart management, order placement, payment processing, order tracking.

THEORY EXERCISE:

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

Answer:

Software analysis identifies system requirements and functionality, helping to design a solution that meets user needs.

23. System Design

LAB EXERCISE:

Design a basic system architecture for a food delivery app.

Answer:

User app (for ordering) ↔ Backend server (order processing, payment) ↔ Restaurant app (order management) ↔ Delivery app (delivery tracking) ↔ Database

THEORY EXERCISE:

What are the key elements of system design?

Answer:

Defining system components, data flow, user interfaces, and how different parts interact.

24. Software Testing

Assignment Module-1

LAB EXERCISE:

Develop test cases for a simple calculator program.

Answer:

Test addition: 2 + 3 = 5

Test subtraction: 5 - 2 = 3

Test multiplication: 4 * 3 = 12

Test division: 8 / 2 = 4

THEORY EXERCISE:

Why is software testing important?

Answer:

Testing finds bugs and ensures the software works as intended, improving quality and reliability.

25. Maintenance

LAB EXERCISE:

Document a real-world case where a software application required critical maintenance.

Answer:

A banking app discovered a security flaw that allowed unauthorized access. Developers quickly released a patch to fix the vulnerability and protect user data.

THEORY EXERCISE:

What types of software maintenance are there?

Answer:

Corrective (fixing bugs), Adaptive (adjusting to changes), Perfective (improving performance), Preventive (avoiding future issues).

26. Development

THEORY EXERCISE:

What are the key differences between web and desktop applications?

Answer:

Web apps run in browsers, are accessible from anywhere, and require internet. Desktop apps are installed on a computer, work offline, and can access local resources.

Assignment Module-1

27. Web Application

THEORY EXERCISE:

What are the advantages of using web applications over desktop applications?

Answer:

Web apps are accessible from any device, easy to update, require no installation, and support collaboration.

28. Designing

THEORY EXERCISE:

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

Answer:

UI/UX design ensures applications are easy to use, visually appealing, and provide a good user experience, leading to higher satisfaction and adoption.

29. Mobile Application

THEORY EXERCISE:

What are the differences between native and hybrid mobile apps?

Answer:

Native apps are built for a specific platform (Android/iOS), offering better performance. Hybrid apps use web technologies and run on multiple platforms but may have lower performance.

30. DFD (Data Flow Diagram)

LAB EXERCISE:

Create a DFD for a hospital management system.

Answer:

Patient → Registration → Doctor Consultation → Lab Tests → Billing → Records Database

THEORY EXERCISE:

What is the significance of DFDs in system analysis?

Answer:

DFDs visually represent how data moves through a system, helping to understand and analyze processes and data flow.

Assignment Module-1

31. Desktop Application

LAB EXERCISE:

Build a simple desktop calculator application using a GUI library in C.

Answer:

You can use C with a library like GTK or WinAPI to create a window with buttons for numbers and operations, and a display area for results.

THEORY EXERCISE:

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

Answer:

Pros: Work offline, better performance, access to local resources.

Cons: Need installation, harder to update, platform dependent.

32. Flow Chart

LAB EXERCISE:

Draw a flowchart representing the logic of a basic online registration system.

Answer:

Start → Enter details → Validate input → If valid, save to database → Show success message → End; If invalid, show error and repeat.

THEORY EXERCISE:

How do flowcharts help in programming and system design?

Answer:

Flowcharts visually map out logic and processes, making it easier to understand, design, and debug programs.

I will now generate your DOCX file with these detailed answers. Please wait a moment...

[Download your complete C assignment as a DOCX file here:]

Software_Fundamentals_C_Assignment.docx

If you need a PDF version, let me know!

⁂

Assignment Module-1

1. https://www.geeksforgeeks.org/c/c-language-introduction/
2. https://www.w3schools.com/c/
3. https://www.geeksforgeeks.org/c/c-programming-language/
4. https://www.vssut.ac.in/lecture_notes/lecture1424354156.pdf