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

ans. A computer program is a sequence or set of instructions in a programming language for a computer to execute. The function of a program is that how it behaves during execution .

---

2. What are the key steps involved in the programming process?

ans. The programming process involves these steps:

a. Understand the Problem: Know what you're solving.
b. Plan: Break the problem into steps (algorithm).
c. Choose Tools: Pick the right language and tools.
d. Write Code: Implement the solution.
e. Test: Check if the program works.
f. Debug: Fix any errors.
g. Optimize: Improve efficiency.
h. Document: Explain how it works.
i. Deploy: Share the program.
j. Maintain: Update and fix as needed.

---

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

ans. low-level programming languages :-

A. require more coding and debugging
B. low-level programming languages are closer to machine code
C. Low-level languages have a more cryptic syntax that is difficult to read and understand

high-level programming languages :-

A. it have a more natural and readable syntax, which makes it easier for programmers to read and understand the code.
B. high-level process map will show the overview of the entire process throughout the organization
C. High-level languages are more abstract and portable

---

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

ans. the client requests information from the server, and the server provides the requested information this is the main roles of client and server in web communication

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

ans. The TCP/IP model is a fundamental framework for computer networking. It stands for Transmission Control Protocol/Internet Protocol, which are the core protocols of the Internet.

It consists of four layers: Link Layer, Internet Layer, Transport Layer, Application Layer

---

6. Explain Client Server Communication

ans. Client-server communication is a request-response mechanism where a client sends a request to a server and the server responds.

---

7. How does broadband differ from fiber-optic internet?

ans. Broadband is a used for high-speed internet generally , while fiber optic is a type of broadband that uses glass or plastic fibers to transmit data fom one place to other

---

8. What are the differences between HTTP and HTTPS protocols?

Ans. HyperText Transfer Protocol (HTTP)

A. It uses a protocol using which hypertext is transferred over the Web
B. Because there are fewer TCP connections, it provides less network congestion.
C. It allows requests and answers to be pipelined via HTTP.
D. It isn't mobile-friendly.
E. It doesn't provide trustworthy exchange (in the absence of retry mechanism).

Hypertext Transfer Protocol Secure (HTTPS)

A. It is an extended version of the Hypertext Transfer Protocol (HTTP). It is used for secure communication.
B. While HTTPS guarantees data security, the HTTP protocol does not provide data security.
C. Increases the visitors' trust to your website.
D. Encrypting and decrypting data across HTTPS connections requires a lot of computation.
E. If there are configuration issues, HTTP will be used by your website to obtain files rather than HTTPS.

---

9. What is the role of encryption in securing applications?

Ans. Encryption is used to protect data from being stolen, changed, or compromised

Encryption protects data in applications by scrambling it into an unreadable format. This prevents unauthorized access to sensitive data, such as financial transactions, messages, and other confidential information

---

10. What is the difference between system software and application software?

Ans. System software

* Manages a computer's hardware, including the memory, processors, and devices
* Runs in the background to maintain the computer's basic functions
* Provides a platform for running application software

Application software

* Performs specific tasks for users, such as word processing, data analysis, and multimedia management
* Improves productivity and efficiency

---

11. What is the significance of modularity in software architecture?

Ans. Modularity is a software design technique that breaks a program into smaller, independent modules.It improves the quality, maintainability, and flexibility of software.

---

12. Why are layers important in software architecture?

Ans. Layers are important in software architecture because they organize the system into distinct levels, each with a specific purpose. This helps in Separation of Concerns and Flexibility and Scalability . In simple words we can say that layers make software systems more organized, adaptable, and easier to work .

---

13. Explain the importance of a development environment in software production.

Ans. Development environment is like a workspace for programmers .It provides the suitable environments and tools or settings for programmers to build a software. it gives tools like text editor . automates the boring tasks for programmers . keeps the project works different so they dont mess up . in simple words we can say that a development environment makes it easier , faster to create a software

---

14. What is the difference between source code and machine code?

Ans. SOURCE CODE : Written in high-level programming languages (eg . Python, Java) that humans can read and understand.

Created by programmers and needs to be translated into machine code.

MACHINE CODE : It is made from binary instructions (0s and 1s) that the computer's processor can execute.

Generated from source code and directly executed by the computer

---

15. Why is version control important in software development?

Ans Version control is important in software development because it helps teams track and manage changes to code, designs, and other project assets . it also allows team to work faster and smarter

---

16. What are the benefits of using Github for students?

Ans. GitHub can be beneficial for students because it provides access to tools and resources for learning and developing software skills

- Includes free access to tools like domain name and hosting, cloud service credits, and learning resources
- Offers free access to developer tools from GitHub's partners
- Allows students to collaborate on projects in private repositories
- Allows students to learn from experienced developers
- Allows students to build a portfolio that showcases their experience

---

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

Ans. Open source Software: Open source software is computer software whose source code is available openly on the internet and programmers can modify it to add new features and capabilities without any cost

Proprietary Software: Proprietary software is computer software where the source codes are publicly not available only the company that has created them can modify it.

---

18. How does GIT improve collaboration in a software development team?

Ans . Git improves collaboration in software development by enabling multiple developers to work on different branches of a project simultaneously. This helps to avoid overwriting each other's work in same project.

---

19. What is the role of application software in businesses?

Ans. Application software helps businesses work more efficiently, make better decisions, and serve customers better. It can automate tasks, improve communication, and help manage data

---

20. What are the main stages of the software development process?

Ans . The main stages of the software development process are.
Planning ,design ,development ,testing ,deployment ,maintenance

---

21. Why is the requirement analysis phase critical in software development?

Ans. The requirement analysis phase is critical in software development because it ensures that the development team and stakeholders understand what needs to be built.

---

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

Ans. Software analysis is a crucial step in the software development process that helps ensure the software meets the needs of the users

---

23. What are the key elements of system design?

Ans. The key elements of system design are:

* Architecture - Defines the system's overall structure and components.
* Data Design - Organizes how data is stored and accessed.
* Interface Design - Specifies interactions between components or users.
* Component Design - Breaks the system into smaller, manageable parts.
* Security - Protects the system from unauthorized access and risks.
* Performance - Ensures the system can handle load and scale as needed.
* Reliability - Makes the system robust and easy to maintain.

---

24. Why is software testing important?

Ans. Software testing is important because

* It helps ensure that software is reliable, secure.

* Software testing reduces project risks related to software quality, security and performance.
* software testing helps to identify and fix issues early in the development process, which can save time

---

25. What types of software maintenance are there?

Ans. There are four main types of software maintenance

* Corrective maintenance
Fixes bugs and errors in software after they are discovered.

* Adaptive maintenance
Modifies software to adapt to changes in the environment, such as new operating systems or regulations.

* Preventive maintenance
Optimizes the system to identify and fix issues before they become problems.

* Perfective maintenance
Improves the software's functionality and performance. This can include adding new features, removing features.

---

26. What are the key differences between web and desktop applications?

Ans. WEB APPLICATION - Web applications are accessed through a browser and can be used on any device with an internet connection

DESKTOP APPLICATION - Desktop applications are installed on a computer and can be used without an internet connection

---

27. What are the advantages of using web applications over desktop applications

Ans. ADVANTAGES OF WEB APPLICATION
* Accessible from anywhere with an internet connection
* Works across different platforms and devices
* Web apps can scale to accommodate many users much more quickly than desktop apps.
* Users can access web apps from Windows PCs, Macs, Linux machines, iPhones, Androids, etc.

---

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

Ans. The full form of UI/UX design is User Interface/User Experience design. It has become very important in the field of application design .The role of a UI/UX designer is to ensure that the application, or software is user-friendly and visually appealing for the user.

---

29. What are the differences between native and hybrid mobile apps?

Ans . NATIVE APPS -

* Native apps are built for a specific operating system.
* example - Java and Kotlin for Android and Swift for ios.

HYBRID MOBILE APPS -

* Hybrid apps are built to work on multiple operating systems
* hybrid apps are easier to develop and maintain.

---

30. What is the significance of DFDs in system analysis?

Ans. DFD refers to Data flow diagrams .They are important in system analysis because they help visualize how data moves through a system. It help identify problems, improve processes, and protect data.

---

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

Ans . PROS -

* Run natively on your computer system and don't need to transfer data over the internet
* With desktop applications, you aren't dependent on an internet connection to use the software
* Users have more control over desktop application software and settings.

CONS -

* You can only access that same app from other devices if you install it separately.
* Manual updates required - Users must manually download and install updates for desktop apps.
* There can also be compatibility issues with different devices, even within the same OS.

---

32. How do flowcharts help in programming and system design?

Ans. Flowcharts help programmers and system designers visualize and organize the steps and decisions in a program or system .In simple words we can say that it helps programmers to develop the most efficient coding because they can clearly see where the data is going at end.

PRACTICAL :-

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

ANS. In c language

```c
#include <stdio.h>
int main() {
    printf("HELLO WORLD");
    return 0;
}
```

```
HELLO WORLD

=== Code Execution Successful ===
```

Structure

- The program is usually saved in a file named hello.c
- The program includes the stdio.h header, which contains the printf function
- The program includes the main function, which contains the printf statement

Syntax

- The syntax to output "HELLO WORLD" is printf("HELLO WORLD")
- The program ends with the statement return 0

In Python

```python
print("HELLO WORLD")
```

```
HELLO WORLD

=== Code Execution Successful ===
```

Structure

- Use a variable to store the string "Hello, World!".
- Call the print() function to display the variable's contents on the screen.
- Use the def keyword to define a function.

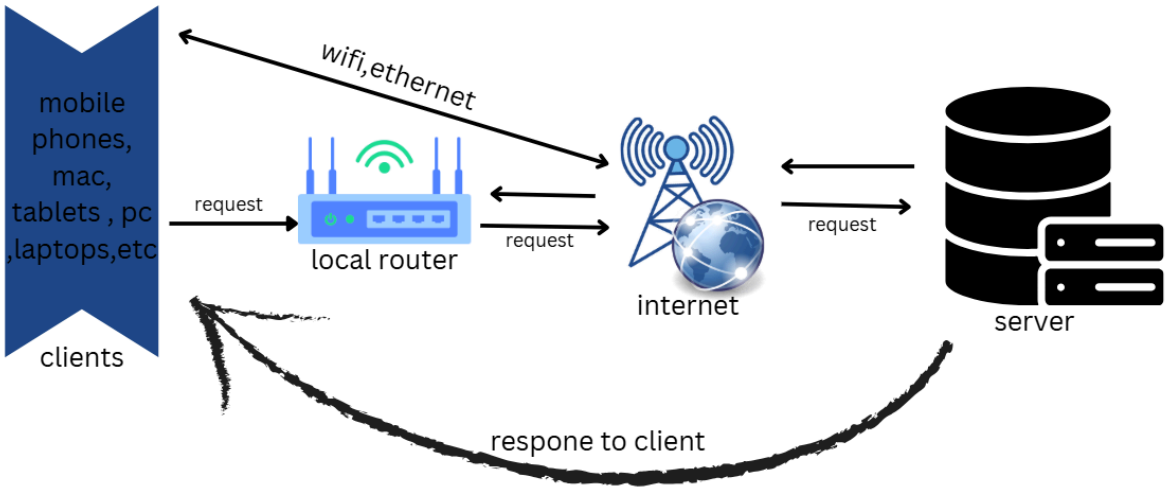
Syntax

- Use double quotation marks to surround the string "Hello, World!".
- Use the print() function to output text.
- Use the print() function to pass none or more expressions separated by commas.

---

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

Ans.



---

3. Design a simple HTTP client-server communication in any language.

Ans. **1. Server (Python)**

The server will listen for incoming HTTP requests on port 8080 and respond with a simple message.

CODE

```
# Simple HTTP Server using http.server module

from http.server import BaseHTTPRequestHandler, HTTPServer

class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        # Respond with HTTP status 200 and a simple message
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(b"Hello, this is the server response!")
```

```
def run(server_class=HTTPServer, handler_class=SimpleHTTPRequestHandler,
port=8080):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    print(f'Starting server on port {port}...')
    httpd.serve_forever()

if __name__ == "__main__":
    run()
```

2. Client (Python)

The client sends a simple GET request to the server and prints the response.

CODE

```
# HTTP Client using requests module

import requests

def send_request():
    url = 'http://localhost:8080'
    response = requests.get(url)

    if response.status_code == 200:
        print("Server Response:", response.text)
    else:
        print(f"Error: {response.status_code}")

if __name__ == "__main__":
    send_request()
```

Expected Output:

- **Server**: The server will print `Starting server on port 8080...` and wait for requests.
- **Client**: The client will print `Server Response: Hello, this is the server response!`.

---

4. Research different types of internet connections (e.g., broadband, fiber, satellite)and list their pros and cons.
Ans. . 1. Fiber-Optic Internet

Pros:

- Extremely fast speeds (up to 1 Gbps +).
- Reliable with low latency.

Cons:

- Limited availability.
- High installation cost.

2. Cable Internet

Pros:

- High speeds (up to 1 Gbps).
- Widely available in urban/suburban areas.

Cons:

- Slower speeds during peak times.
- More expensive than DSL.

3. DSL (Digital Subscriber Line)

Pros:

- Affordable.
- Available in most areas.

Cons:

- Slower speeds (typically up to 100 Mbps).
- Speed drops with distance from the provider

4. Satellite Internet

Pros:

- Available in remote/rural areas.
- Quick setup with no need for infrastructure.

Cons:

- High latency.
- Weather can affect performance.

5. Mobile Broadband (4G/5G)

Pros:

- Wireless and portable.
- Fast speeds (especially with 5G).

Cons:

- Limited data (especially 4G).
- Data caps and potential throttling.

6. Fixed Wireless Internet

Pros:

- Fast installation.
- Good for rural areas.

Cons:

- Limited availability (requires line-of-sight).
- Can be affected by weather.

7. Ethernet (Wired)

Pros:

- Stable and reliable.
- No interference from external factors.

Cons:

- Limited mobility (requires a physical connection).
- Requires installation of cables.

---

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

Ans. .

HTTP request :-

- To make an HTTP GET request to a server, you can use the following command:

```
curl http://example.com
```

- To make an HTTP POST request with JSON data, you can use:

```
curl -X POST -H "Content-Type: application/json" -d '{"key":"value"}'
http://example.com/api
```

FTP Request :-

- To download a file from an FTP server, you can use:

```
curl -T file.txt ftp://ftp.example.com/upload/
```

- To upload a file to an FTP server, you can use:

```
curl ftp://ftp.example.com/file.txt -o file.txt
```

---

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

Ans. 1. Injection Attacks

Attackers inject malicious input (e.g., SQL, command, or NoSQL injection) to manipulate application behavior or access data.
Solution:

- Use parameterized queries or prepared statements.
- Sanitize and validate user inputs.
- Regularly update databases and code libraries.

2. Cross-Site Scripting (XSS)

Attackers inject malicious scripts into web pages to steal data or impersonate users.
Solution:

- Encode user inputs before rendering.
- Implement Content Security Policy (CSP).
- Use input sanitization tools like DOMPurify.

3. Broken Authentication

Weak authentication allows unauthorized access (e.g., weak passwords or token exposure).
Solution:

- Enforce strong passwords and use MFA.
- Secure session tokens in HTTP-only cookies.
- Regularly test and improve authentication mechanisms.

---

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

Ans. **1.** System Software

System software manages and supports the operation of computer hardware and provides the platform for running application software.

- Operating System:
  - Windows 10 / 11
  - macOS
  - Linux (Ubuntu)
- Device Drivers:
  - Graphics driver (NVIDIA GeForce)
  - Audio driver (Realtek)
- Firmware:

- BIOS/UEFI (Basic Input/Output System)

Application software

- Word processing software
  Allows users to create, edit, and format documents. Examples include Microsoft Word, Google Docs, and Apple Pages.
- Web browsers
  Allow users to access and navigate the internet. Examples include Google Chrome, Microsoft Edge, Safari, Firefox, and Brave.
- Music software
  Allows users to listen to music. Examples include Pandora, Apple Music, and Spotify.
- Communication software
  Allows users to communicate with others. Examples include Slack, Skype, Zoom, and Teams.
- Games
  Entertainment software that users can play.

---

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

Ans. OP

```
| Presentation     |      <-- User Interface (Web, Mobile)
| Layer (Client)   |          Where the user interacts with the web application
| (Browser)        |
+------------------+
        |
        v
+------------------+
| Application      |      <-- Business Logic (API, Server-side Logic)
| Layer (Server)   |        Responsible for processing the logic, serving content, and
| (Web Server,     |              handling requests from the client
|  Application     |
|  Framework)      |
+------------------+
        |
        v
+------------------+
| Data Layer       |      <-- Database (Data Storage)
| (Database)       |       The backend database or data storage, where all the
| (MySQL, MongoDB) |   information is stored, retrieved, and manipulated.
+------------------+
```

9. Create a case study on the functionality of the presentation, business logic, and Data access layers of a given software system.

Ans. Case Study: Functionality of Presentation, Business Logic, and Data Access Layers

1. Presentation Layer

Purpose: Provides the user interface for interactions.
Key Features:

- User Interface: Web pages or mobile app for students, librarians, and admins.
- Functions:
    - Students: Search for books, view availability, and reserve books.
    - Librarians: Add new books, update book details, and track overdue books.
    - Admins: Manage users and system settings.
- Technology: HTML, CSS, JavaScript, or React.

Example:
A student searches for a book titled *"Data Structures"*. The interface displays the search results with availability status.

2. Business Logic Layer

Purpose: Processes user requests and applies business rules.
Key Features**:**

- Handles book search logic (e.g., keyword matching).
- Manages borrow/return policies (e.g., max borrow limit, due dates).
- Ensures user permissions (e.g., only admins can delete records).

Example:
When a student reserves a book, the system checks:

1. If the student has overdue books.
2. If the book is already reserved by someone else.
3. Applies reservation policies.

3. Data Access Layer

Purpose: Handles interactions with the database.
Key Features:

- Performs CRUD operations (Create, Read, Update, Delete).
- Secures data access through APIs or queries.
- Optimizes database performance.

Example:
When a librarian adds a new book:

1. The system sends a query to the database to insert book details.
2. Ensures data consistency by validating fields like ISBN and author.

---

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

Ans. **Exploring Software Environments**

Software environments are structured configurations in which applications or software systems are developed, tested, and deployed. Each type of environment has a specific purpose:

1. **Development Environment**:
    - **Purpose**: Used by developers to write and test code locally.
    - **Characteristics**:
        - Includes tools like Integrated Development Environments (IDEs), compilers, and debuggers.
        - Typically unstable, as active development occurs here.
        - May include mock services or dummy data for initial testing.
2. **Testing Environment**:
    - **Purpose**: Used to test the software for bugs, performance, and integration issues.
    - **Characteristics**:
        - Simulates production-like conditions.
        - Often includes automated testing frameworks.
        - Test data may be sanitized but similar to production data.
        - Used for various types of testing: unit, integration, system, and user acceptance.
3. **Production Environment**:
    - **Purpose**: Where the software is deployed for end-users.

- **Characteristics**:
  - Must be highly stable, secure, and optimized.
  - Uses real user data.
  - Downtime or issues in this environment directly impact users.

Setting Up a Basic Environment in a Virtual Machine

Prerequisites

- A computer with virtualization enabled in BIOS/UEFI.
- Virtualization software such as **VirtualBox**, **VMware**, or **Hyper-V**.
- An ISO image or installation file for the operating system (e.g., Ubuntu, Windows).

Steps to Set Up a Virtual Machine

1. **Install Virtualization Software**:
   - Download and install software like VirtualBox or VMware.
2. **Create a Virtual Machine**:
   - Open the virtualization software and create a new virtual machine.
   - Configure:
     - **Name**: Give a descriptive name (e.g., "DevEnv").
     - **OS Type**: Select the operating system.
     - **Memory**: Allocate sufficient RAM (e.g., 2-4 GB for Linux, more for Windows).
     - **Storage**: Create a virtual hard disk (e.g., 20 GB or more).
3. **Install the Operating System**:
   - Attach the ISO file to the virtual machine as a bootable disk.
   - Start the virtual machine and follow the OS installation process.
4. **Configure the Environment**:
   - **Development Environment**:
     - Install IDEs (e.g., VS Code, PyCharm).
     - Set up version control (e.g., Git).
     - Add programming language runtimes (e.g., Python, Node.js).
   - **Testing Environment**:
     - Install testing tools (e.g., Selenium, JUnit).
     - Configure a local web server or database for integration testing.
   - **Production Environment**:
     - Install web servers (e.g., Nginx, Apache).

        - Secure the system (e.g., firewall, SSH keys).
5. **Snapshot the VM**:
    - Take a snapshot of the VM once it's set up. This allows you to restore it to a clean state if needed.

---

11. Write and upload your firstsource code file to Github.

ANS. project hover

Html https://github.com/dhyeykamani/demo/blob/main/index.html

Css https://github.com/dhyeykamani/demo/blob/main/style.css

---

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

ANS Github repository link

https://github.com/dhyeykamani/demo

. Push Changes to GitHub:

Push your changes:
git push origin main

You've created a repository, committed changes, and pushed them to GitHub.

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

Ans.

---

14. Create a list of software you use regularly and classify them into the Following categories: system, application, and utility software.

Ans. Here’s a list of software I regularly use, categorized into System, Application, and Utility software:

1. System Software

System software manages and supports the operation of computer hardware and provides the platform for running application software.

- Operating System:
  - Windows 10 / 11
  - macOS
  - Linux (Ubuntu)
- Device Drivers:
  - Graphics driver (NVIDIA GeForce)
  - Audio driver (Realtek)
- Firmware:
  - BIOS/UEFI (Basic Input/Output System)

2. Application Software

Application software is designed for end-users to perform specific tasks or activities, often related to business, productivity, or entertainment.

- Web Browsers:
  - Google Chrome
  - Mozilla Firefox
  - Microsoft Edge
- Productivity Tools:
  - Microsoft Office (Word, Excel, PowerPoint)
  - Google Workspace (Docs, Sheets, Slides)
  - Notion (note-taking and task management)
- Communication:
  - Microsoft Teams
  - Slack
- Media Players:
  - VLC Media Player
  - Spotify
- Cloud Storage & File Management:
  - Google Drive
  - Dropbox

3. Utility Software

Utility software helps manage, maintain, and protect the system, offering support and optimization functions.

- Antivirus Software:
    - Windows Defender
    - McAfee
- Backup and Recovery:
    - Acronis True Image
    - Time Machine (macOS)
- Disk Management:
    - CCleaner
    - Disk Cleanup (Windows)
- File Synchronization:
    - SyncBack (for backup and file synchronization)
    - FreeFileSync
- System Monitoring:
    - Task Manager (Windows)
    - Activity Monitor (macOS)

---

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

Ans **1. Cloning a Repository**

Cloning is the process of creating a local copy of a repository from GitHub.

Step 1: Find a Repository to Clone

Let's assume you're cloning a repository from GitHub. For example, you can use this demo repository.

Step 2: Clone the Repository

1. Go to the repository page on GitHub.

2. Click on the green "Code" button, and copy the repository URL.

   It will look something like this:
   `https://github.com/octocat/Hello-World.git`

Open your terminal or Git Bash and run:

git clone https://github.com/octocat/Hello-World.git

3.

Navigate into the cloned repository directory:

cd Hello-World

3. Branching

Branches are used to work on different features without affecting the main code.

Step 1: Create a New Branch

Create a new branch where you'll make your changes:

git checkout -b my-feature-branch

This command:

- Creates a new branch called `my-feature-branch`.
- Switches you to that branch.

Step 2: Make Changes

Now, make changes in any file within the repository. For example, open the `README.md` file and add a new line like:

This is my new feature.

Step 3: Stage and Commit Changes

Stage the changes (mark files for committing):

git add README.md

1.

Commit your changes with a descriptive message:

git commit -m "Added a new feature in README"

5. Merging Changes

Step 1: Switch Back to the Main Branch

Before merging, ensure you're on the `main` branch (or the default branch):

git checkout main

Step 2: Pull the Latest Changes

To make sure you're up to date with the latest changes from the repository, run:

git pull origin main

Step 3: Merge the Feature Branch

Now, merge your feature branch (`my-feature-branch`) into the `main` branch:

git merge my-feature-branch

If there are no conflicts, this will successfully merge your changes.

Step 4: Push the Merged Changes

Push the updated `main` branch with the merged changes to GitHub:

git push origin main

---

16. Write a report on the various types of application software and how they Improve productivity

Ans. Application software consists of programs designed to perform specific user tasks efficiently. This report briefly highlights different types of application software and their impact on productivity.

Types of Application Software

Productivity Software

- Examples: Microsoft Office, Google Workspace.
- Role: Helps create documents, spreadsheets, and presentations quickly and accurately.

Communication Software

- Examples: Zoom, Slack.
- Role: Enables instant communication and collaboration, especially for remote work.

Business Software

- Examples: SAP, Salesforce.
- Role: Automates business operations like inventory, CRM, and accounting.

Educational Software

- Examples: Duolingo, Moodle.
- Role: Facilitates online learning and skill development.

Creative Software

- Examples: Adobe Photoshop, Canva.
- Role: Assists in designing and creating multimedia content efficiently.

Utility Software

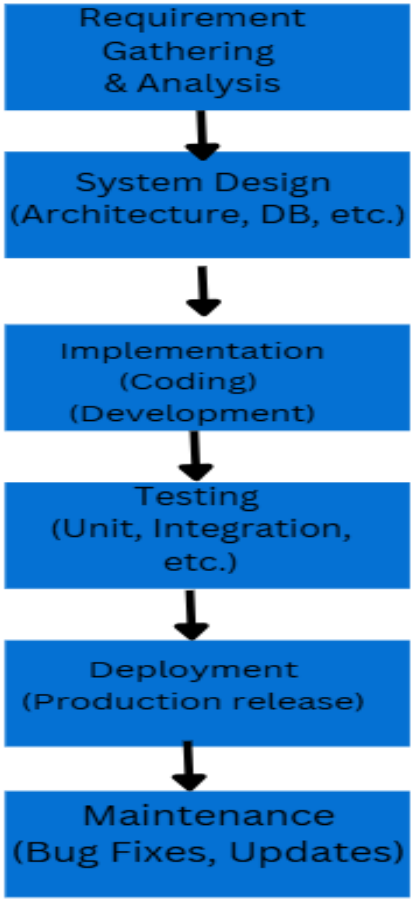
- Examples: Antivirus, backup tools.
- Role: Enhances system performance and ensures data security.

How Application Software Improves Productivity

- Automation: Reduces time spent on repetitive tasks.
- Collaboration: Tools enable real-time teamwork.
- Flexibility: Cloud apps provide access from anywhere.
- Error Reduction: Features like error detection improve output quality.
- Data Insights: Business tools generate actionable analytics.

---

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



---

18. Write a requirement specification for a simple library management system

Ans.**Introduction**

The **Library Management System (LMS)** is a software system designed to manage books, members, transactions (book loans/returns), and generate reports. It helps automate library operations, improving productivity for librarians and members.

Functional Requirements

1. **Book Management**:
    - **Add, Remove, Update** books (title, author, genre, ISBN, number of copies).
    - **View Book List** with filtering and sorting.
2. **Member Management**:
    - **Register and Manage** members (name, contact, membership type).

    - **Update Member Info** and view all members.
3. **Transaction Management**:
    - **Issue and Return Books** with dates.
    - **Track Overdue Books** and notify users.
4. **Search Functionality**:
    - **Search Books** by title, author, ISBN, genre.
    - **Advanced Search** with filters.
5. **Reporting**:
    - Generate reports on books, member borrowing, overdue books, etc.
    - **Export Reports** to PDF/Excel.
6. **User Roles**:
    - **Admin (Librarian)**: Full access to all features.
    - **Member (User)**: Access to book information, borrowing history.

Non-Functional Requirements

1. **Performance**: Handle 500 members and 1000 books without lag.
2. **Reliability**: Ensure data integrity and 99.5% system availability.
3. **Usability**: Simple, intuitive UI for all users.
4. **Security**: User authentication and secure data storage.
5. **Compatibility**: Web-based, mobile-responsive, compatible with modern browsers.
6. **Backup & Recovery**: Regular backups and data recovery options.

System Features

1. **UI**:
    - **Login Screen** for admins and members.
    - **Dashboards** for both roles with quick access to key features.
2. **Search & Filter**:Simple search bar for books, with filter options.
3. **Notifications**:
    - Automated overdue notifications to members via email.

---

19. Perform a functional analysis for an online shopping system.

Ans. A functional analysis for an online shopping system involves identifying and documenting the system's functions and how they work together. This analysis can help you identify missing requirements and improve the user experience.

Steps for performing a functional analysis

- Identify functional requirements: Specify how orders are processed, including registration and checkout.
- Document the functions: Use diagrams like use case diagrams, flowcharts, or mind maps to document the interactions between different entities.
- Test the functions: Use software testing to verify that the product meets quality criteria.
- Analyze user interactions: Use session recording software to see how users interact with your site.
- Evaluate the functions: Use a requirements engineering framework, quality attributes, or SMART criteria to evaluate the functions.

Functional requirements examples

- User registration and login
- Password recovery
- Email verification
- Security questions
- Order statuses
- B2B order management

Functional requirements for online stores speed, ease of use, accessibility, mobile-friendliness, and personalization.

---

20. Design a basic system architecture for a food delivery app.

Ans. Basic System Architecture for a Food Delivery App

1. Client Layer

- **Mobile App**: For customers, restaurants, and delivery agents.
- **Web App**: For restaurant partners and admin dashboard.

2. Application Layer (Backend)

- **User Management**: Handles authentication, profiles, and roles (customer, restaurant, delivery agent).
- **Order Management**: Processes order placement, tracking, and history.
- **Restaurant Management**: Manages menu, availability, and pricing.
- **Delivery Management**: Assigns delivery agents, tracks locations, and updates delivery status.
- **Payment Gateway**: Secure payment processing and refunds.

3. Data Layer (Database)

- **User Database**: Stores user details and roles.

- **Order Database**: Tracks orders, status, and history.
- **Restaurant Database**: Stores menu, pricing, and restaurant info.
- **Delivery Database**: Tracks agent details and delivery routes.

4. External Services

- **Payment Gateway**: For online payments (e.g., Stripe, PayPal).
- **Notification Service**: SMS, email, and push notifications.
- **Maps API**: For real-time location tracking and route optimization.

5. Infrastructure Layer

- **Server**: For backend services (e.g., AWS, Azure).
- **CDN**: For fast content delivery (e.g., images, menus).
- **Load Balancer**: Distributes traffic evenly across servers.
- **Cloud Storage**: Stores images, logs, and backups.

Flow

1. **Customer**: Places an order → **Backend** → Restaurant & Delivery notified.
2. **Restaurant**: Prepares food → Notifies completion.
3. **Delivery Agent**: Picks up order → Delivers to customer using maps.
4. **Backend**: Updates order status and sends notifications.

---

21. Develop test cases for a simple calculator program.

Ans. **Test Cases for Simple Calculator**

1. Addition

- Input: `2 + 3` → Output: `5`
- Input: `-5 + 3` → Output: `-2`

2. Subtraction

- Input: `7 - 3` → Output: `4`
- Input: `-2 - 5` → Output: `-7`

3. Multiplication

- Input: `4 * 5` → Output: `20`
- Input: `0 * 8` → Output: `0`

4. Division

- Input: `10 / 2` → Output: `5`
- Input: `5 / 0` → Output: Error (Division by zero)

5. Modulus

- Input: `10 % 3` → Output: `1`
- Input: `7 % 0` → Output: Error (Modulo by zero)

6. Edge Cases

- Input: `Large Number + 1` → Output: Overflow/Error
- Input: `Non-numeric Input` → Output: Error (Invalid input)

7. Mixed Operations

- Input: `(2 + 3) * 4` → Output: `20`
- Input: `10 / (5 - 5)` → Output: Error (Division by zero)

---

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

Ans **Real-World Case: Equifax Data Breach (2017)**

Background

Equifax, one of the largest credit reporting agencies in the United States, suffered a major data breach in 2017 that exposed the personal information of approximately 147 million individuals. The breach is considered one of the largest and most damaging in history, largely due to the sensitivity of the data involved, such as Social Security numbers, birth dates, addresses, and more.

Cause of the Incident

The breach was traced to a vulnerability in **Apache Struts**, an open-source framework used by Equifax in one of its web applications. The vulnerability, **CVE-2017-5638**, had been disclosed publicly on March 7, 2017, and a patch was made available at the same time. However, Equifax failed to apply the patch to its systems in a timely manner.

Impact

The breach had severe consequences:

- **Financial Cost:** Equifax agreed to pay at least $575 million as part of a settlement with the Federal Trade Commission (FTC), state attorneys general, and other entities.
- **Reputation Damage:** The company's reputation suffered significantly, with widespread public criticism and loss of trust.
- **Operational Fallout:** Executives, including the CEO, CIO, and CSO, stepped down following the incident.

Critical Maintenance Required

The case highlights the importance of timely **patch management** in software maintenance. Equifax's failure to:

1. Identify and prioritize the vulnerability.
2. Apply the patch across all affected systems.
3. Test their systems for other unpatched or vulnerable components.

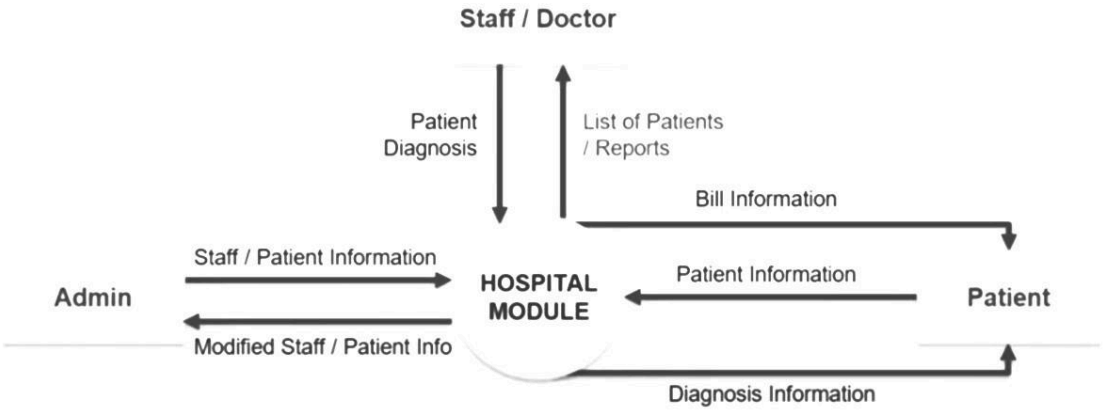
Lessons Learned

1. **Regular Patch Management:** Organizations must implement a robust patch management process to address vulnerabilities promptly.
2. **Vulnerability Scanning:** Continuous scanning and monitoring of systems for known vulnerabilities can help detect issues proactively.
3. **Incident Response Plans:** Organizations should have well-documented and rehearsed incident response plans to mitigate damage quickly in case of a breach.
4. **Accountability:** Clear accountability and communication within IT and security teams are critical for addressing vulnerabilities.

The Equifax breach remains a stark reminder of the importance of critical maintenance in software systems to ensure the security and integrity of sensitive data.

---

23 Create a DFD for a hospital management system.



---

24. Build a simple desktop calculator application using a GUI library

Ans code in python is given below for desktop calculator application

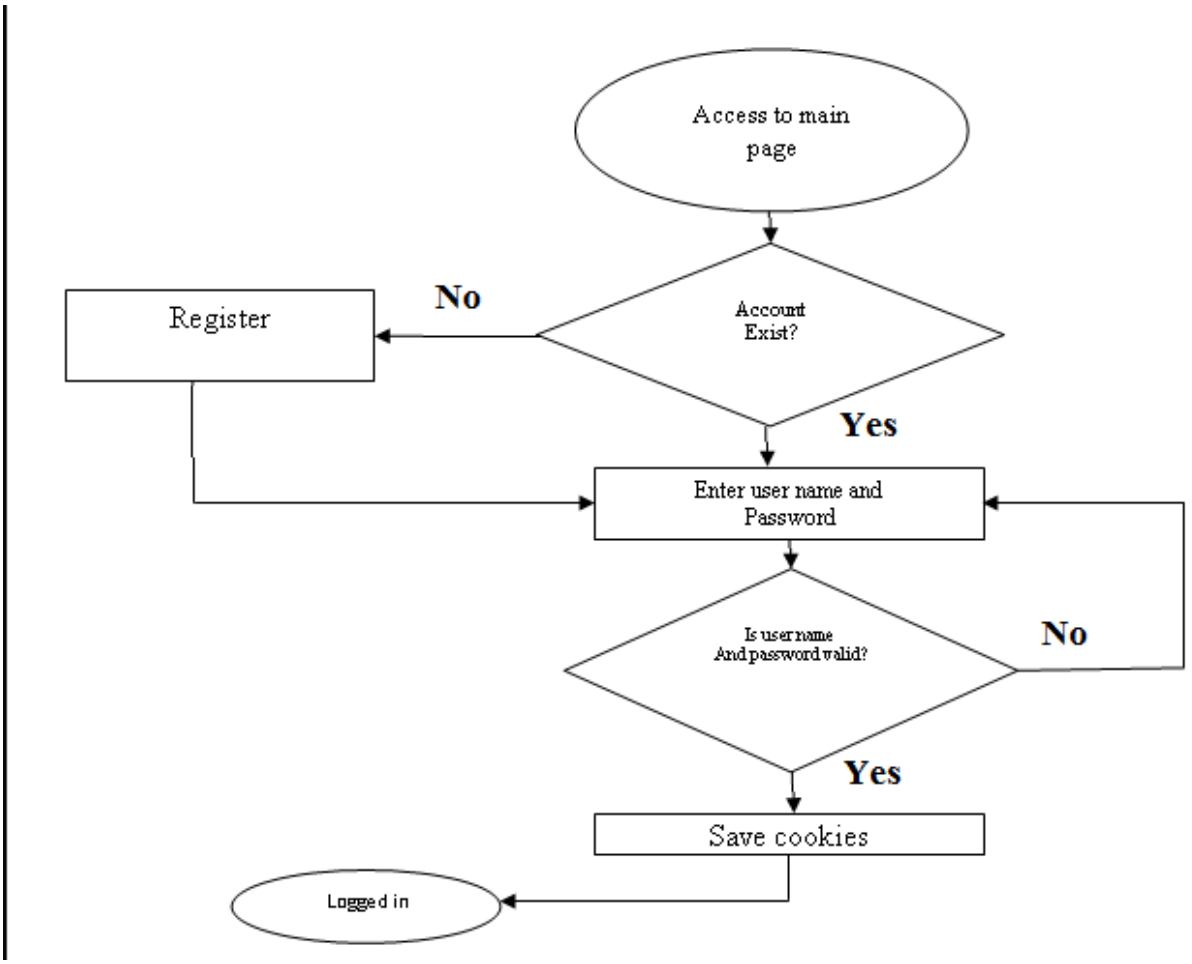
```
from tkinter import *

def button_click(number):
    """Handles button clicks."""
    current = entry.get()
    entry.delete(0, END)
    entry.insert(0, str(current) + str(number))

def button_clear():
    """Clears the entry field."""
    entry.delete(0, END)

def button_equal():
    """Performs the calculation."""
    try:
        result = eval(entry.get())
        entry.delete(0, END)
        entry.insert(0, str(result))
    except:
        entry.delete(0, END)
        entry.insert(0, "Error")

# Create the main window
window = Tk()
window.title("Simple Calculator")
```

```
# Create the entry field
entry = Entry(window, width=35, borderwidth=5)
entry.grid(row=0, column=0, columnspan=4, padx=10, pady=10)

# Define buttons
buttons = [
    '7', '8', '9', '/',
    '4', '5', '6', '*',
    '1', '2', '3', '-',
    '0', '.', '=', '+'
]

row_index = 1
col_index = 0

for button in buttons:
    button_text = Button(window, text=button, padx=40, pady=20, command=lambda
b=button: button_click(b))
    button_text.grid(row=row_index, column=col_index)
    col_index += 1
    if col_index > 3:
        col_index = 0
        row_index += 1

# Create clear button
button_clear = Button(window, text="Clear", padx=79, pady=20,
command=button_clear)
button_clear.grid(row=5, column=0, columnspan=2)

# Create equal button
button_equal = Button(window, text="=", padx=81, pady=20,
command=button_equal)
button_equal.grid(row=5, column=2, columnspan=2)

window.mainloop()
```

---

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

Ans.


Access to main page
Account Exist?
No
Register
Yes
Enter user name and Password
Is user name And password valid?
No
Yes
Save cookies
Logged in