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

**Ans.** 1.Hello world program in C :

#include<stdio.h>

void main()

{

printf("Hello World\n");

}

2. Hello World program in C++ :

#include<iostream>

int main()

{

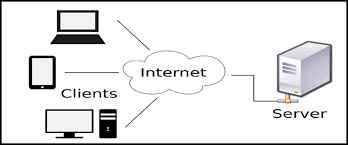
std:: cout<<"hello world"<<std::endl;

}

* **C:** Uses #include <stdio.h> to include the standard input/output library, which provides functions like printf().
* **C++:** Uses #include <iostream> to include the input/output stream library, which provides objects like std::cout and std::endl.

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

**Ans. Client – Server communication:**



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

**Ans. Server-side (server.py):**

import http.server  
import socketserver  
  
PORT = 8000  
  
class MyHandler(http.server.SimpleHTTPRequestHandler):  
 def do\_GET(self):  
 self.send\_response(200)  
 self.send\_header("Content-type", "text/html")  
 self.end\_headers()  
 self.wfile.write(b"<h1>Hello from the server!</h1>")  
  
with socketserver.TCPServer(("", PORT), MyHandler) as httpd:  
 print(f"Serving at port {PORT}")  
 httpd.serve\_forever()

**Client-side (client.py):**

import http.client  
  
HOST = "localhost"  
PORT = 8000  
  
conn = http.client.HTTPConnection(HOST, PORT)  
conn.request("GET", "/")  
response = conn.getresponse()  
  
print(f"Status: {response.status}")  
print(f"Reason: {response.reason}")  
print(f"Data: {response.read().decode('utf-8')}")  
  
conn.close()

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

**Ans.** The Internet is a global network of computers connected. There are several types of internet connections.

**1.Dial-Up Connection:**

**Pros:** It is low cost and easy to set up.

**Cons:** It is Slow speed and outdated technology.

**2.Broadband connection:**

**Pros:** It ishighspeed, always-on connectivity, consistency, affordability, and wide availability.

**Cons:** High cost, location dependant, limited in rural areas.

**3.Digital Subscriber Line:**

**Pros:** It is affordable, use existing infrastructure, always on.

**Cons:** Slower in speed, distance-sensitive, limited availability.

**4.Fiber:**

**Pros:** Superior speed, high reliability, scalability, improve security.

**Cons:** Higher cost, limited availability, specialised installation.

**5.Satellite Connection:**

**Pros:** Wide Availability, fast installation, good for basic needs.

**Cons:** High latency, weather interference, expensive.

**6.Wireless Connection:**

**Pros:** Convenience & Mobility**,** easy setup, scalability.

**Cons:** Interference, security risk, limited range.

**7. Cellular Connection:**

**Pros:** Portability and flexibility, easy setup, broad availability, lower initial costs.

**Cons:** Inconsistent speed and reliability, high cost.

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

**Ans.** Simulating HTTP Requests with curl:

1. **GET Request:**

To retrieve content from a URL:

“ curl <http://example.com/path/to/resource>”

1. **POST Request:**

To send data to a server:

“curl -X POST -H "Content-Type: application/json" -d '{"key": "value"}' <http://example.com/api/endpoint>”

Simulating FTP Requests with curl:

**1.Uploading a File:**

curl -u "username:password" -T /local/path/to/file.txt ftp://ftp.example.com/remote/directory/

**2.Downloading a File:**

curl -u "username:password" -o /local/path/to/save/file.txt <ftp://ftp.example.com/remote/directory/file.txt>

Simulating HTTP Requests with wget:

wget is primarily used for downloading files from the web.

**1. Downloading a File:**

wget http://example.com/path/to/file.zip

**2. Downloading a Website Recursively:**

wget -r -l 1 http://example.com/

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

**Ans.** Three common application security vulnerabilities are Injection attacks, Broken Authentication, and Security Misconfigurations.

**1. Injection Attacks**

* Explanation:

An attacker inserts malicious data into an application that it then interprets as a command, rather than data. A common example is [SQL Injection](https://www.google.com/search?sca_esv=c0ca25e8fd581d08&cs=0&sxsrf=AE3TifPjg3IKXpwVpMx6DdoEpnPDMhxGhQ%3A1757596141944&q=SQL+Injection&sa=X&ved=2ahUKEwjQu4zt49CPAxWvXGcHHfbUACMQxccNegQIDxAB&mstk=AUtExfBUWMriHVO291F3t4toEMq5LZ2bw2qsv9yoKn9QfiNUa74EjZ5GMJdht1b_m3lJuUbQFR6OCUzL7-qO5uJ1YqjEUzV0ZTcu3GCm0PikGGOlLT3hfp5rdvzVn7HBzJt7lq_8mnbdfB8sOZSfpA66U9w4ge7qo7FCnX8utzIL0v-QzLY&csui=3), where an attacker injects SQL code into an input field to manipulate the application's database.

* Solution:

Implement input validation and sanitization to ensure that user input is treated as data only and not as executable code. Also, utilize parameterized queries for database interactions, which separate code from data.

**2. Broken Authentication**

* Explanation:

Weaknesses in authentication, or the process of verifying a user's identity, and session management allow attackers to compromise passwords, keys, or session tokens and impersonate legitimate users.

* Solution:

Enforce strong authentication mechanisms, such as multi-factor authentication (MFA) and complex password requirements. Implement robust session management practices, including proper session expiration and protection against hijacking.

**3. Security Misconfiguration**

* Explanation:

This vulnerability stems from insecurely configured default settings or overly permissive security settings. Examples include using default credentials on a system, leaving debug modes enabled in production, or not changing default usernames and passwords.

* Solution:

Harden all system configurations by removing or disabling default credentials and unnecessary features. Regularly patch and update all software, and ensure that security headers and other security features are properly configured and enabled

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

**Ans. 1.Word Processing Software:**Word processing software can be classified as a type of Application Software or General-Purpose Application Software that performs specific tasks, such as creating, editing, and formatting text-based documents like letters, reports, and books.

**2.Presentation Software:** Presentation software can be classified by its platform (desktop-based, web-based, mobile), its format (traditional slide-based, non-linear, whiteboard-style, video-based, infographic-focused), its features (AI-powered, interactive, collaborative), or its purpose (business, educational, training, marketing).

**3.Multimedia Software:** Multimedia software can be classified by its functionality into categories like graphics/image editors (e.g., Adobe Photoshop), audio editors (e.g., Audacity), video editors (e.g., Adobe Premiere Pro), animation/motion graphics software (e.g., Adobe After Effects), presentation software (e.g., Microsoft PowerPoint), authoring tools (to create interactive multimedia), and media players (e.g., VLC Media Player).

**4.Web Browser:** Web browsers can be classified by their user interface (graphical vs. text-based), by their primary function or focus (e.g., privacy, speed, mobile), or by their underlying layout engine (e.g., Blink, Gecko, WebKit). The most common browsers are graphical, like Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge, but text-based browsers also exist for specific needs.

**5.Graphic Software:** Graphic software is classified primarily by its output: Raster Graphics Software manipulates pixel-based images (like photos), Vector Graphics Software creates scalable images from mathematical lines and shapes (like logos), and 3D Graphics Software generates three-dimensional objects and animations. Other categories include Desktop Publishing Software for page layout, [UI/UX Design Software](https://www.google.com/search?sca_esv=29c8be3fd3fde57f&cs=0&sxsrf=AE3TifOWNU5LP6ZPOY3FVEAC6cVoqI9MpQ%3A1757598239909&q=UI%2FUX+Design+Software&sa=X&ved=2ahUKEwiI5czV69CPAxWP6zgGHTk9F5UQxccNegQIBBAB&mstk=AUtExfChCwNTCpShRb-7yC1djauvpymWzV54piplLu3YFGr1HhBoLLNI6i555c0FCiZNJ4vMEVxYKK9OETHJ54jfDg1hwBOWrCVwFlME85okdSX1SzwE5fWPe0XWMaaSdPetYBhnIsMzn2VsIJw2rbzA4qoVgef9mKw6RV_P3elsdh72PnrtN7spIdl0bj7Wv2TPnbQTqHDF7tZ-rWHCFj55GuXEezhvNYiv4u9r_KnX3a2Hh-AmPtqki-Qsxd75jDx2Oc_3Gzvri-oKRyjVJnClMUDZsmiGC_rk3fwQ5zNLACC_yQ&csui=3) for digital interfaces, and Special-Purpose Software like photo editors or 3D modelers.

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

**Ans.**



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

**Ans. 1. Presentation Layer (User Interface):**

* Functionality:

This layer is responsible for displaying information to the user and capturing user input. It handles the visual aspects of the application and facilitates user interaction.

**2. Business Logic Layer (Application Logic):**

* Functionality:

This layer encapsulates the core business rules and operations of the system. It processes user requests, applies business rules, and orchestrates interactions with the data access layer.

**3. Data Access Layer (Persistence):**

* Functionality:

This layer is responsible for abstracting the underlying data storage mechanism. It handles all interactions with the database, including retrieving, storing, updating, and deleting data.

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

**Ans.** Software Environments: Development, Testing, and Production

Software development typically involves distinct environments tailored to specific stages of the software lifecycle:

* **Development Environment:**

This is where developers write, debug, and test code. It's often a local setup on a developer's machine, providing flexibility and rapid iteration.

* **Testing Environment:**

Also known as QA (Quality Assurance) or Staging, this environment closely mirrors the production environment to identify bugs and ensure functionality, performance, and security before release.

* **Production Environment:**

This is the live environment where the software is deployed and made accessible to end-users.

**Setting Up a Basic Environment in a Virtual Machine**

A virtual machine (VM) provides an isolated environment for setting up and experimenting with different software configurations without affecting the host system. Here's how to set up a basic development environment in a VM:

* **Install Virtualization Software:**

Choose a hypervisor like Oracle VirtualBox, VMware Workstation Player, or Hyper-V (if using Windows Pro/Enterprise).

* **Create a New Virtual Machine:**
  + Open your chosen virtualization software.
  + Initiate the process to create a new VM, providing a name, type (e.g., Linux, Windows), and version of the operating system you intend to install.
  + Allocate RAM and hard disk space for the VM. A dynamically allocated disk is often convenient.
* **Install an Operating System:**
  + Mount an ISO image of your desired operating system (e.g., Ubuntu, CentOS, Windows) to the VM's virtual CD/DVD drive.
  + Start the VM and follow the on-screen instructions to install the OS.
* **Install Development Tools:** 
  + Once the OS is installed and running, install necessary development tools within the VM. This might include:
    - Package Manager: apt (Debian/Ubuntu), yum/dnf (Red Hat/CentOS)
    - Version Control: Git
    - Programming Language Runtime: Python, Node.js, Java, etc.
    - IDE/Code Editor: VS Code, IntelliJ IDEA, Eclipse
    - Database: PostgreSQL, MySQL, MongoDB
* **Configure Network (Optional but Recommended):**
  + Set up network adapters within the VM settings to allow internet access and potentially communication with your host machine. Bridged or NAT modes are common choices.

**Que. Write and upload your first source code file to Github.**

**Ans.** File Name : “File1.c”

Source code:

#include<stdio.h>

Void main()

{

Printf(“Hello World”);

}

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

**Ans. System software:**

These software programs are designed to run a computer's application programs and hardware.

Examples:

* [Operating Systems](https://www.google.com/search?sca_esv=83105bac496dfc01&sxsrf=AE3TifNEwHqnt3ELvLb2BYDKmctXIHsuAw%3A1757607361248&q=Operating+Systems&sa=X&ved=2ahUKEwjY36_TjdGPAxU1oa8BHUD8K3sQxccNegQIKRAB&mstk=AUtExfD9zAaAOydG3pwTE02IBxgtireMCUnuWggzw2uIi2Hw9C-wd6jgbjo5tXdzAOZudyxAx3gCNiXJM_6bXWzz0EIa7jYqwDdXawbe5LJwTQHIKAF6jyOZy5Z4iDBHWDfAF7iVjQaS8sJ9nYMH9Q21j4l7ogiky1Mn1tFz-z807baxacal_KT79erfg9IC9BJKBVXZGvajsIaPJrkVJx88VKKcv59GhZFLHXkHp2glVLh_h5XL4Z022Cj91eY4xLO8retXJQA8rqlbzjyC43P89TFyDn_iYKV2XnPd4Jj5lJdlJw&csui=3): Microsoft Windows, macOS, Linux (like Ubuntu), and Android.
* Device Drivers: Audio drivers, Video drivers.
* Utility Programs: antivirus software.

**Application Software:**

The most common type of software, application software is a computer software package that performs a specific function for a user, or in some cases, for another application.

Examples:

* Web Browsers**:** Google Chrome, Mozilla Firefox, Apple Safari.
* Word Processors**:** Microsoft Word, for creating documents.
* Spreadsheet Programs**:** Microsoft Excel, for calculations and data analysis.
* Media Players**:** Software for listening to music or watching videos, like Spotify or VLC Media Player.
* Games: A wide variety of entertainment applications.
* Database Management Systems (DBMS**):** Like Microsoft Access or Oracle, used for storing and organizing data.

**Utility Software:**

Utility software provides functions to help users manage, maintain, and optimize a computer system, performing specific tasks that enhance its performance, functionality, and security.

Examples: Antivirus Software, Backup Software**.**

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

**Ans. 1. Cloning a Repository:**

* Find a repository:

Locate a suitable Git repository, for example, a public repository on GitHub or Bitbucket, or create a new one.

* Copy the URL:

Navigate to the repository's page and find the "Code" or "Clone" button. Copy the HTTPS or SSH URL provided.

* Clone the repository:

Open your terminal or command prompt and navigate to the desired directory where you want to store the local copy. Execute the following command, replacing [repository\_url] with the copied URL:

Code:

git clone [repository\_url]

* Navigate into the repository**:** Change your current directory to the newly cloned repository:

Code:

cd [repository\_name]

**2. Creating and Switching Branches:**

* Create a new branch**:** Create a new branch named feature-branch (or any other descriptive name):

Code:

git branch feature-branch

* Switch to the new branch**:** Move your working directory to the feature-branch:

Code:

git checkout feature-branch

Alternatively, you can create and switch to a new branch in a single command:

Code:

git checkout -b feature-branch

**3. Merging Branches:**

* Switch to the target branch**:** Switch back to the main or master branch (or the branch you want to merge into):

Code:

git checkout main

* Merge the feature branch**:** Merge the changes from feature-branch into the main branch:

Code:

git merge feature-branch

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

**Ans. Types of Application Software**

* Productivity Software**:**
  + Word Processors: Applications like [Microsoft Word](https://www.google.com/search?sca_esv=83105bac496dfc01&cs=0&sxsrf=AE3TifPwFupvikANi5BHxQvwrfFM5ff5CA%3A1757610065987&q=Microsoft+Word&sa=X&ved=2ahUKEwiZw77cl9GPAxWz4jgGHZYpJ9kQxccNegQIORAB&mstk=AUtExfBFdclsxHxS99VtmwTsYWmXREunDDAZquuzOP4nYrgDcCYBAaaLTtJUdgL9VdhKRFRkn0MxCZ2uDwXsGcSlbNZ_onnnTkqaHqtWaDP80uGY4XRr9cDR8lLjzJvaI-Q73fIOarnR5tfwWrkqb4UqgSeNRTXQMBs0WoFaXY9vlahmKOCDCag_f8WP9ipi72M1Sp9koqL7RX4zrCHFEAjMaxIgqiDoitt2tWMvq1xryWDvC6AiXPCT-u52fyxpfrTvyfT9dWZxIGXIHoOnnddDMxaAaEi--1sccyW3btMWXSy_RQ&csui=3) and Google Docs.
  + Spreadsheets: Software such as [Microsoft Excel](https://www.google.com/search?sca_esv=83105bac496dfc01&cs=0&sxsrf=AE3TifPwFupvikANi5BHxQvwrfFM5ff5CA%3A1757610065987&q=Microsoft+Excel&sa=X&ved=2ahUKEwiZw77cl9GPAxWz4jgGHZYpJ9kQxccNegQIMBAB&mstk=AUtExfBFdclsxHxS99VtmwTsYWmXREunDDAZquuzOP4nYrgDcCYBAaaLTtJUdgL9VdhKRFRkn0MxCZ2uDwXsGcSlbNZ_onnnTkqaHqtWaDP80uGY4XRr9cDR8lLjzJvaI-Q73fIOarnR5tfwWrkqb4UqgSeNRTXQMBs0WoFaXY9vlahmKOCDCag_f8WP9ipi72M1Sp9koqL7RX4zrCHFEAjMaxIgqiDoitt2tWMvq1xryWDvC6AiXPCT-u52fyxpfrTvyfT9dWZxIGXIHoOnnddDMxaAaEi--1sccyW3btMWXSy_RQ&csui=3) and Google Sheets.
  + Presentation Software: Tools like Google Slides and PowerPoint.
* Graphics and Media Software:
  + Graphic Design Software: Programs like Adobe Photoshop.
  + Video/Audio Editing Software: These tools are essential for professionals to edit and produce high-quality video and audio content**.**
* Data Management Software:
  + Database Software: Applications that allow users to organize, store, and manage large sets of data, enabling easier retrieval and use of information.
  + Customer Relationship Management (CRM) Software: Used to manage customer interactions and data, building stronger relationships and improving customer service.
* Enterprise Software:
  + ERP Systems: Integrated suites that manage various business processes, from finance to human resources, providing a centralized view of operations.
  + Project Management Software: Tools that help teams plan, track, and execute projects efficiently, improving coordination and workflow.
* Web Browsers: Applications like Chrome and Firefox.

**How Application Software Improves Productivity**

* Automation:

Software automates repetitive and time-consuming tasks, freeing up employees to focus on strategic activities and increasing overall output.

* Streamlined Workflows:

By providing structured and integrated tools, application software helps to organize and optimize business processes, reducing inefficiencies.

* Enhanced Collaboration:

Cloud-based platforms and communication tools allow teams to collaborate in real time, regardless of their physical location, fostering teamwork.

* Data-Driven Decision-Making:

Business intelligence and analytics tools within application software provide access to real-time data and insights, empowering businesses to make informed, strategic decisions.

* Increased Efficiency:

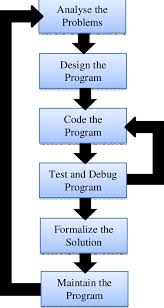
The specific functionalities of application software help users perform tasks more quickly and accurately than manual methods, leading to higher overall efficiency.

* Cost Reduction:

Automating tasks and improving resource allocation through application software can significantly lower operational costs associated with manual labor.

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

**Ans.**



**Que. Write a requirement specification for a simple library management system.**

**Ans. Functional Requirements:**

* Item Management:
  + Add, update, and remove book information (title, author, ISBN, genre, etc.).
  + Track item status (available, checked out, etc.).
  + Categorize items.
* User Management:
  + Register and manage user (patron) accounts and profiles.
  + Set different user roles and permissions.
* Circulation:
  + Issue books to members, recording due dates.
  + Process book returns and update inventory.
  + Allow users to renew borrowed books.
* Fines & Fees:
  + Calculate and collect fines for overdue books.
  + Notify users and librarians about overdue items.
* Reporting & Analytics:
  + Generate reports on book circulation, popular items, and user activity.
  + Provide inventory status reports.
* Search:
  + Allow users and librarians to search for books by title, author, or ISBN.
* Acquisition:
  + Support ordering, receiving, and invoicing of new library materials.
* Interlibrary Loan:
  + Enable the request and handling of materials from other libraries.

**Non-Functional Requirements:**

* Usability: The system should be user-friendly and easy to use for both librarians and members.
* Reliability: The system must be accurate and reliable in its operations.
* Security: Implement different user access levels to ensure data security and protect personal information.
* Performance:
  + Respond quickly to requests.
  + Handle multiple users and transactions efficiently.
* Availability: The system needs to be accessible 24/7, often via the web.
* Maintainability: Allow for easy updates and modifications.
* Scalability: Support multiple locations and users, especially in larger institutions.

**Hardware & Software Requirements:**

* Operating System: Windows, Linux, or macOS.
* Database: A relational database system like MySQL or PostgreSQL.
* Server: Powerful server with sufficient RAM and storage for the database and user transactions.
* Client Devices: Standard computers with keyboards and mice for internal use, and potentially mobile devices for external access.
* Networking: Internet connectivity for web-based access.

**Que. Perform a functional analysis for an online shopping system.**

**Ans.**

* Define User Flows:

Map out the typical steps a user takes to complete a task, like finding and buying a product.

* Identify Key Entities:

Determine the main components of the system, such as users, products, carts, and orders.

* Use Diagrams:

Employ tools like UML use case diagrams or flowcharts to visualize interactions and process flows between entities.

* Prioritize Requirements:

Use techniques like the [MoSCoW method](https://www.google.com/search?sca_esv=cb1e697b99ad7555&cs=0&sxsrf=AE3TifPej9kN5vSx2MM_O2jBvzt_ICuHmQ%3A1757611945435&q=MoSCoW+method&sa=X&ved=2ahUKEwjsmdXcntGPAxUNwzgGHbvMDTMQxccNegQIfhAB&mstk=AUtExfDR6_T9GPh7qaJNo-d9wLnXiDUUHvilFERoHxTnf75UC2y4gCJu6zSlUUtCqpO6Ykto4cw2VroA9c7bxwOcf3gsVNFCX6N14GkekWZylKUAu87HgxyqojidBM0pHxdfHVqZMHFHF_g-8O3hQoFyagMR673WMg08pagfAVdmZtYFniw&csui=3" \t "_blank) (Must have, Should have, Could have, Won't have) to prioritize features for development.

* Test Thoroughly:

After analysis, conduct functional testing to ensure that the system functions as specified.

**Que. Design a basic system architecture for a food delivery app.**

**Ans.**

**Que. Develop test cases for a simple calculator program.**

**Ans. 1. Basic Functionality Tests:**

* Addition:
  + Add two positive integers (e.g., 5 + 3 = 8).
  + Add a positive and a negative integer (e.g., 7 + (-2) = 5).
  + Add two negative integers (e.g., -4 + (-6) = -10).
  + Add zero to a number (e.g., 10 + 0 = 10).
* Subtraction:
  + Subtract a smaller positive integer from a larger positive integer (e.g., 9 - 4 = 5).
  + Subtract a larger positive integer from a smaller positive integer (e.g., 3 - 8 = -5).
  + Subtract a negative integer (e.g., 5 - (-3) = 8).
  + Subtract zero from a number (e.g., 12 - 0 = 12).
* Multiplication:
  + Multiply two positive integers (e.g., 6 \* 2 = 12).
  + Multiply a positive and a negative integer (e.g., 4 \* (-5) = -20).
  + Multiply two negative integers (e.g., -3 \* (-7) = 21).
  + Multiply by zero (e.g., 15 \* 0 = 0).
  + Multiply by one (e.g., 20 \* 1 = 20).
* Division:
  + Divide two positive integers with a whole number result (e.g., 10 / 2 = 5).
  + Divide two positive integers with a decimal result (e.g., 7 / 2 = 3.5).
  + Divide a positive by a negative integer (e.g., 12 / (-3) = -4).
  + Divide a negative by a positive integer (e.g., -18 / 6 = -3).
  + Divide a negative by a negative integer (e.g., -20 / (-5) = 4).
  + Divide zero by a non-zero number (e.g., 0 / 5 = 0).

**2. Edge Case Tests:**

* Division by Zero:

Attempt to divide any number by zero (e.g., 10 / 0). The expected result is an error message or "Infinity".

* Large Numbers:

Perform operations with very large numbers to check for overflow or precision issues.

* Decimal Numbers:

Perform all operations with various decimal numbers, including those with many decimal places.

* Mixed Operations:

Test a sequence of operations (e.g., 5 + 3 \* 2). If the calculator follows order of operations (BODMAS/PEMDAS), the result should be 11. If it processes sequentially, it would be 16.

* Invalid Input:
  + Enter non-numeric characters as operands.
  + Enter an invalid operator.

**3. User Interface/Interaction Tests (if applicable):**

* Clear/Reset Functionality: Verify that the "Clear" or "C" button correctly resets the display and any stored values.
* Backspace/Delete Functionality: Verify that a backspace or delete button removes the last entered digit.
* Display Limits: Check how the calculator handles results that exceed the display capacity.

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

**Ans.** The Problem

* What Happened:

In the early 2000s, NASA's Mars Odyssey orbiter began showing signs of navigational instability.

* The Cause:

The software responsible for the orbiter's attitude control was found to have a subtle bug that caused errors in its orientation and movement.

* The Impact:

These errors led to potentially serious navigational inaccuracies, which could have jeopardized the mission.

The Maintenance Action

* Solution:

NASA's Jet Propulsion Laboratory (JPL) developed and deployed a software patch.

* Method:

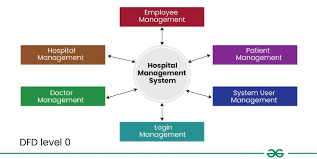
The software fix was transmitted wirelessly from Earth to the Mars Odyssey orbiter.

* Outcome:

The successful application of the patch corrected the navigational errors, restoring the orbiter's stability and ensuring the mission's continued success.

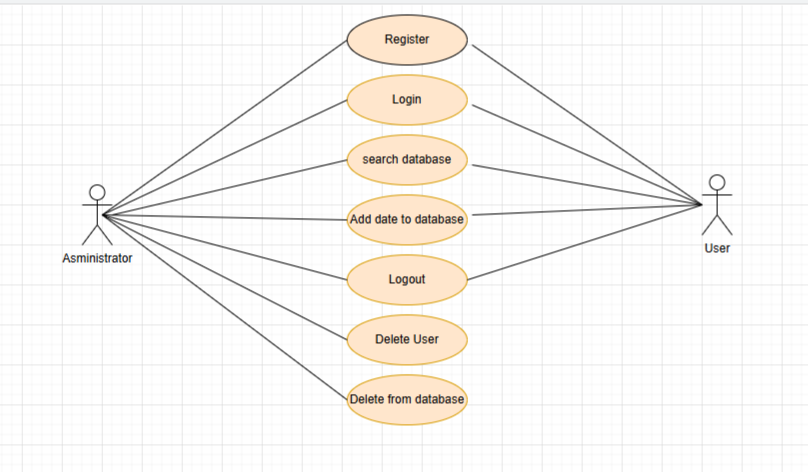
**Que. Create a DFD for a hospital management system.**

**Ans.**



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

**Ans.**

****