Q-1 Research and provide three real-world applications where C programming is extensively used, such as in embedded systems, operating systems, or game development

* **1. Embedded Systems**
* **Examples:**
* Microcontrollers in home appliances (microwaves, washing machines)
* Automotive systems (anti-lock braking systems, engine control units)
* IoT devices and smart sensors
* **Why C is used:**
* Direct hardware manipulation
* Low memory footprint
* Real-time performance requirements
* Portability across various hardware platforms
* **Real-World Use Case:**  
  **Automotive ECU (Electronic Control Unit)** – Automotive manufacturers like **Bosch** and **Continental** use C to program the firmware of ECUs that control braking, engine timing, and airbags.
* **2. Operating Systems**
* **Examples:**
* UNIX and Linux kernels
* Windows kernel components
* Android low-level system components
* **Why C is used:**
* Close-to-metal programming allows efficient resource management
* Direct access to memory and hardware
* Portability and performance
* **Real-World Use Case:**  
  **Linux Kernel** – The Linux kernel, used in everything from smartphones to servers, is predominantly written in C. Major tech companies like **Google**, **IBM**, and **Red Hat** contribute to and rely on Linux-based systems.
* **3. Game Development**
* **Examples:**

Game engines (like Doom engine, Quake engine)

Performance-critical components in games

Console firmware and SDKs

* **Why C is used:**

High performance and low-level memory management

Compatibility with hardware and game consoles

Tight control over CPU and GPU usage

* **Real-World Use Case:**  
  **Id Software's Doom and Quake Engines** – Early 3D game engines were written in C for performance reasons. These engines laid the foundation for many modern 3D gaming techniques.

Q-2 Install a C compiler on your system and configure the IDE. Write your first program to print "Hello, World!" and run it. 3. Basic Structure of a C Program

* Step 1: Install a C Compiler
* Choose your operating system below:
* Windows: Install GCC via MinGW
* Download MinGW:
* Visit: https://sourceforge.net/projects/mingw/
* Install Components:
* During setup, make sure to select:

mingw32-gcc-g++

mingw32-base

mingw32-gcc-objc

* Add to PATH:

Add C:\MinGW\bin to your system’s environment variable PATH.

* Linux (Ubuntu/Debian):
* Open a terminal and run:

bash

sudo apt update

sudo apt install build-essential

* macOS:

Install Xcode Command Line Tools:

bash

xcode-select --install

* Step 2: Choose and Configure an IDE

Recommended IDEs:

Code::Blocks – All-in-one installer for GCC + IDE

* Visual Studio Code – Lightweight and customizable
* Dev-C++ – Simple, beginner-friendly (Windows)
* Example: Set Up Visual Studio Code (VS Code)
* Download and install VS Code: https://code.visualstudio.com
* Install the C/C++ extension by Microsoft.
* Set up tasks.json and launch.json to build and run your code (you can ask me if you need help setting that up).
* Step 3: Write and Save Your First Program
* Create a new file called hello.c and write:

#include <stdio.h>

int main() {

printf("Hello, World!\n");

return 0;

}

* Step 4: Compile and Run
* Using Command Line:
* In the terminal or command prompt:
* bash

gcc hello.c -o hello

./hello # On Linux/macOS

hello.exe # On Windows (or just `hello` if in PowerShell)

* Output:
* Hello, World!

Q-3 Write a C program that includes variables, constants, and comments. Declare and use different data types (int, char, float) and display their values.

* #include <stdio.h>

int main() {

// Constant declaration

const float PI = 3.14159;

// Variable declarations

int age = 25; // Integer variable

char grade = 'A'; // Character variable

float height = 5.9; // Floating-point variable

// Displaying values

printf("Age (int): %d\n", age);

printf("Grade (char): %c\n", grade);

printf("Height (float): %.1f\n", height);

printf("Value of PI (const float): %.5f\n", PI);

// End of program

return 0;

}

Q-4 Write a C program that accepts two integers from the user and performs arithmetic, relational, and logical operations on them. Display the results.

* #include <stdio.h>

int main() {

int a, b;

// Prompt user to enter two integers

printf("Enter first integer: ");

scanf("%d", &a);

printf("Enter second integer: ");

scanf("%d", &b);

// Arithmetic operations

printf("\n--- Arithmetic Operations ---\n");

printf("a + b = %d\n", a + b);

printf("a - b = %d\n", a - b);

printf("a \* b = %d\n", a \* b);

if (b != 0) {

printf("a / b = %d\n", a / b); // Integer division

printf("a %% b = %d\n", a % b); // Remainder (modulus)

} else {

printf("Division and modulus by zero are undefined.\n");

}

// Relational operations

printf("\n--- Relational Operations ---\n");

printf("a == b: %d\n", a == b);

printf("a != b: %d\n", a != b);

printf("a > b: %d\n", a > b);

printf("a < b: %d\n", a < b);

printf("a >= b: %d\n", a >= b);

printf("a <= b: %d\n", a <= b);

// Logical operations

printf("\n--- Logical Operations ---\n");

printf("(a && b): %d\n", (a && b)); // True if both are non-zero

printf("(a || b): %d\n", (a || b)); // True if at least one is non-zero

printf("!(a): %d\n", !a); // True if a is zero

printf("!(b): %d\n", !b); // True if b is zero

return 0;

}

* Output

Enter first integer: 10

Enter second integer: 5

--- Arithmetic Operations ---

a + b = 15

a - b = 5

a \* b = 50

a / b = 2

a % b = 0

--- Relational Operations ---

a == b: 0

a != b: 1

a > b: 1

a < b: 0

a >= b: 1

a <= b: 0

--- Logical Operations ---

(a && b): 1

(a || b): 1

!(a): 0

!(b): 0

Q-5 Write a C program to print numbers from 1 to 10 using all three types of loops (while, for, do-while).

* #include <stdio.h>

int main() {

int i;

printf("Using while loop:\n");

i = 1;

while (i <= 10) {

printf("%d ", i);

i++;

}

printf("\n");

printf("Using for loop:\n");

for (i = 1; i <= 10; i++) {

printf("%d ", i);

}

printf("\n");

printf("Using do-while loop:\n");

i = 1;

do {

printf("%d ", i);

i++;

} while (i <= 10);

printf("\n");

}

* Output
* Using while loop:

1 2 3 4 5 6 7 8 9 10

* Using for loop:

1 2 3 4 5 6 7 8 9 10

* Using do-while loop:

1 2 3 4 5 6 7 8 9 10

Q-6 Write a C program that uses the break statement to stop printing numbers when it reaches 5. Modify the program to skip printing the number 3 using the continue statement.

* #include <stdio.h>

int main() {

int i;

for (i = 1; i <= 10; i++) {

if (i == 3) {

continue; // Skip printing number 3

}

if (i == 5) {

break; // Stop the loop when i reaches 5

}

printf("%d\n", i);

}

}

* Output
* 1
* 2
* 4

Q-7 Write a C program that calculates the factorial of a number using a function. Include function declaration, definition, and call.

* #include <stdio.h>

// Function declaration

int factorial(int n);

int main() {

int num;

printf("Enter a positive integer: ");

scanf("%d", &num);

if (num < 0) {

printf("Factorial is not defined for negative numbers.\n");

} else {

// Function call

int result = factorial(num);

printf("Factorial of %d = %d\n", num, result);

}

}

// Function definition

int factorial(int n) {

int fact = 1;

for (int i = 1; i <= n; i++) {

fact \*= i;

}

}

Q-8 Write a C program that stores 5 integers in a one-dimensional array and prints them. Extend this to handle a two-dimensional array (3x3 matrix) and calculate the sum of all elements.

* #include <stdio.h>

int main() {

// Part 1: 1D array

int arr1D[5] = {10, 20, 30, 40, 50};

printf("1D Array Elements:\n");

for (int i = 0; i < 5; i++) {

printf("%d ", arr1D[i]);

}

printf("\n");

// Part 2: 2D array (3x3 matrix)

int arr2D[3][3] = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

int sum = 0;

printf("\n2D Array Elements (3x3 Matrix):\n");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

printf("%d ", arr2D[i][j]);

sum += arr2D[i][j];

}

printf("\n");

}

printf("\nSum of all elements in the 2D array: %d\n", sum);

return 0;

}

Output:

* 1D Array Elements:
* 10 20 30 40 50
* 2D Array Elements (3x3 Matrix):
* 1 2 3
* 4 5 6
* 7 8 9
* Sum of all elements in the 2D array: 45

Q-9 Write a C program that takes two strings from the user and concatenates them using strcat(). Display the concatenated string and its length using strlen().

* #include <stdio.h>

#include <string.h>

int main() {

char str1[100], str2[100];

// Input two strings from user

printf("Enter the first string: ");

fgets(str1, sizeof(str1), stdin);

str1[strcspn(str1, "\n")] = '\0'; // Remove trailing newline

printf("Enter the second string: ");

fgets(str2, sizeof(str2), stdin);

str2[strcspn(str2, "\n")] = '\0'; // Remove trailing newline

// Concatenate str2 to str1

strcat(str1, str2);

// Display concatenated string and its length

printf("Concatenated string: %s\n", str1);

printf("Length of concatenated string: %lu\n", strlen(str1));

return 0;

}

* Output:
* Enter the first string: Hello
* Enter the second string: World
* Concatenated string: HelloWorld
* Length of concatenated string: 10