**Module 2: Introduction to Programming**

# **1.Overview of C Programming:-**

**THEORY EXERCISE: Write an essay covering the history and evolution of C programming. Explain its importance and why it is still used today.**

**Answer:-**

**History and Importance of C Programming:-**

**The C programming language was developed in the early 1970s by Dennis Ritchie at Bell Labs. It evolved from earlier languages like BCPL and B. In 1973, the UNIX operating system was rewritten in C, which made the language very popular. Later, C was standardized through versions like K&R C (1978), ANSI C (1989), C99, and C11.**

**C influenced many modern languages such as C++, Java, C#, and JavaScript, and its syntax became the foundation of modern programming.**

### **Why C is important and still used:-**

* **Many operating systems are written in C (Windows, Linux, macOS).**
* **It is widely used in embedded systems, microcontrollers, and IoT devices.**
* **C is very fast and gives direct control over hardware.**
* **C programs are highly portable across different machines.**
* **It remains essential for performance-critical systems, game engines, and databases.**

**LAB EXERCISE: Research and provide three real-world applications where C programming is extensively used, such as in embedded systems, operating systems, or game development.**

**Answer:Three Real-World Applications of C Programming:-**

### **1. Operating Systems**

**C is used to build major operating systems like Windows, Linux, and macOS. It provides low-level memory access and high performance, making it ideal for kernel and system-level programming.**

### **2. Embedded Systems**

**C is widely used in microcontrollers, automotive systems, medical devices, and IoT devices. It allows direct control of hardware with efficient memory usage, which is essential for small embedded machines.**

### **3. Game Development & Game Engines**

**Many game engines (like parts of Unity, Unreal Engine, and custom engines) are built using C/C++. C provides speed and control needed for rendering, physics, and real-time game performance.**

# **2. Setting Up Environment :-**

**THEORY EXERCISE: Describe the steps to install a C compiler (e.g., GCC) and set up an Integrated Development Environment (IDE) like DevC++, VS Code, or CodeBlocks.**

**Answer:Steps to Install a C Compiler and Set Up an IDE:**

## **1. Installing GCC Compiler (Windows using MinGW):-**

1. **Go to the MinGW download page and install MinGW Setup.**
2. **Open the installer → Select mingw32-gcc-g++ from the package list.**
3. **Click Apply Changes to install the GCC compiler.**
4. **After installation, copy the MinGW bin path (example: C:\MinGW\bin).**
5. **Add this path to Environment Variables → Path, so GCC works from anywhere.**

**Open Command Prompt and type:  
  
 gcc --version**

1. **If it shows a version, the compiler is installed successfully.**

# **2. Setting Up an IDE:-**

## **Option A: Dev-C++**

1. **Download Dev-C++ from its official site.**
2. **Install and open it.**
3. **Create a New Project → Console Application → C Language.**
4. **Write your program and press Compile & Run.  
    (Dev-C++ comes with GCC preinstalled.)**

## **Option B: VS Code:-**

1. **Install Visual Studio Code from its official website.**
2. **Open VS Code → Go to Extensions.**
3. **Install:**
   * **C/C++ Extension by Microsoft**
   * **Code Runner (optional for quick running)**
4. **Make sure GCC is already installed (via MinGW).**
5. **Create a folder, open it in VS Code.**
6. **Make a new file: program.c.**
7. **To run:**
   * **Terminal → Run Build Task**
   * **or use Code Runner.**

## **Option C: CodeBlocks:-**

1. **Download CodeBlocks with MinGW bundle (important—compiler included).**
2. **Install and open CodeBlocks.**
3. **Go to File → New → Project → Console Application.**
4. **Choose C, write your program, and click Build & Run.**

**LAB EXERCISE: Install a C compiler on your system and configure the IDE. Write your first program to print "Hello, World!" and run it.**

**Answer:- Install a C Compiler and Configure the IDE:-**

* **Install MinGW or TDM-GCC to get the GCC compiler.**
* **Set the compiler path in Environment Variables (e.g., C:\MinGW\bin).**
* **Install an IDE such as Dev-C++, CodeBlocks, or VS Code.**
* **Configure the IDE to use the GCC compiler (CodeBlocks/Dev-C++ usually detect it automatically).**

**Create a new C file and type the following code:-**

**#include <stdio.h>**

**int main() {**

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

**return 0;**

**}**

**Output:-**

**Hello,world!**

**3. Basic Structure of a C Program:-**

**THEORY EXERCISE: Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples.**

**Answer:- A C program is made up of several important parts such as header files, main function, comments, data types, and variables. Each part plays a specific role in how the program works.**

## **1. Header Files:-**

## **Headers contain built-in functions that your program can use. Example:#include<stdio.h>**

## **2. Comments:-**

**Comments explain the code. They are ignored by the compiler.**

* **Single-line comment: // This is a comment**
* **Multi-line comment: /\* This is a multi-line comment \*/**

## **3. main() Function:-**

## **Every C program must have a main function. Execution starts from here.**

## **int main() {**

**return 0;**

**}**

## **4. Data Types:-**

**Data types define what kind of value a variable can store.**

**Common C data types:**

* **int – stores integers**
* **float – decimal numbers**
* **char – single character**
* **double – large decimal values**

**Example:**

**int age;**

**float price;**

**char letter;**

## **5. Variables:-**

**Variables are names used to store values in memory.**

**Example:**

**int num = 10;**

**float salary = 25000.50;**

**char grade = 'A';**

**Example program:-**

**#include<stdio.h>**

**Int main(){**

**Int a=10;**

**Int b=5.5;**

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

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

**printf("Value of b = %.1f", b);**

**Return 0;**

**}**

**Output:-**

**Hello, World!**

**Value of a = 10**

**Value of b = 5.5**

**LAB EXERCISE: Write a C program that includes variables, constants, and comments. Declare and use different data types (int, char, float) and display their values.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**const int maxMarks = 100;**

**int roll = 23;**

**float height = 5.7;**

**char section = 'B';**

**printf("Max Marks: %d\n", maxMarks);**

**printf("Roll Number: %d\n", roll);**

**printf("Height: %f\n", height);**

**printf("Section: %c\n", section);**

**return 0;**

**}**

**output:-**

**Max Marks: 100**

**Roll Number: 23**

**Height: 5.7000**

**Section: B**

**4. Operators in C:-**

**THEORY EXERCISE: Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators.**

**Answer:-**

**1.Arithmetic operator:-**

**used for mathematical calculation.**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| **+** | **Addition** | **a + b** |
| **-** | **Subtraction** | **a - b** |
| **\*** | **Multiplication** | **a \* b** |
| **/** | **Division** | **a / b** |
| **%** | **Modulus (remainder)** | **a % b** |

**2.Relational operators:-**

**Used for compare two values.**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| **==** | **Equal to** | **a == b** |
| **!=** | **Not equal to** | **a != b** |
| **>** | **Greater than** | **a > b** |
| **<** | **Less than** | **a < b** |
| **>=** | **Greater than or equal to** | **a >= b** |
| **<=** | **Less than or equal to** | **a <= b** |

**3.Logical operators:-**

**Used for combined multiple conditions.**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| **&&** | **Logical AND** | **(a > 0 && b > 0)** |
| **!** | **Logical NOT** | **!(a > b)** |

**4.Assignment operator:-**

**Used to assign value to variables.**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Meaning** |
| **=** | **a = 10** | **Assigns 10 to variable a** |
| **+=** | **a += 5** | **Same as a = a + 5** |
| **-=** | **a -= 3** | **Same as a = a - 3** |
| **\*=** | **a \*= 2** | **Same as a = a \* 2** |
| **/=** | **a /= 4** | **Same as a = a / 4** |
| **%=** | **a %= 3** | **Same as a = a % 3** |

**5.increment and decrement operators:-**

**used to increase or decrease a variable by 1.**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| **++a** | **Pre-increment** | **Increase first, then use value** |
| **a++** | **Post-increment** | **Use value, then increase** |
| **--a** | **Pre-decrement** | **Decrease first** |
| **a--** | **Post-decrement** | **Use, then decrease** |

**6.Bitwise operators:-**

**Used to perform operations at the bit level.**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| **&** | **Bitwise AND** | **a & b** |
| **&** | **Bitwise AND** | **Bitwise OR** |
| **^** | **Bitwise XOR** | **a ^ b** |
| **~** | **Bitwise NOT** | **~a** |
| **<<** | **Left shift** | **a << 1** |
| **>>** | **Right shift** | **a >> 1** |

**LAB EXERCISE: Write a C program that accepts two integers from the user and performs arithmetic, relational, and logical operations on them. Display the results.**

**Answer:**

**#include <stdio.h>**

**int main() {**

**int a, b;**

**printf("Enter first number: ");**

**scanf("%d", &a);**

**printf("Enter second number: ");**

**scanf("%d", &b);**

**printf("\n--- Arithmetic Results ---\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("\n--- Relational Results ---\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("\n--- Logical Results ---\n");**

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

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

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

**return 0;**

**}**

**Output:-**

**--- Arithmetic Results ---**

**a + b = 9**

**a - b = 3**

**a \* b = 18**

**a / b = 2**

**a % b = 0**

**--- Relational Results ---**

**a > b = 1**

**a < b = 0**

**a == b = 0**

**a != b = 1**

**--- Logical Results ---**

**(a && b) = 1**

**(a || b) = 1**

**!(a) = 0**

**5. Control Flow Statements in C:-**

**THEORY EXERCISE:**

**Explain decision-making statements in C (if, else, nested if-else, switch).**

**Provide examples of each.**

**Answer:-Decision-making statements allow a program to choose different actions based on conditions.**

## **1. if Statement:-**

**Used to run a block of code only when the condition is true.**

**Example:**

**if (a > 0) {**

**printf("Number is positive");**

**}**

## **2. if–else Statement:-**

**Runs one block if condition is true, otherwise runs the other block.**

**Example:**

**if (age >= 18) {**

**printf("Eligible to vote");**

**} else {**

**printf("Not eligible");**

**}**

## **3. Nested if–else:-**

## **An if or else block inside another if–else.Used when multiple conditions must be checked.**

**Example:**

**if (marks >= 90) {**

**printf("Grade A");**

**} else {**

**if (marks >= 60) {**

**printf("Grade B");**

**} else {**

**printf("Grade C");**

**}**

**}**

## **4. switch Statement:-**

**Used when you want to check multiple fixed values of the same variable.**

**Example:-**

**switch (day) {**

**case 1: printf("Monday"); break;**

**case 2: printf("Tuesday"); break;**

**case 3: printf("Wednesday"); break;**

**default: printf("Invalid day");**

**}**

**LAB EXERCISE:- Write a C program to check if a number is even or odd using an if-else statement. Extend the program using a switch statement to display the month name based on the user’s input (1 for January, 2 for February, etc.).**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int num, month;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**if (num % 2 == 0) {**

**printf("%d is Even\n", num);**

**} else {**

**printf("%d is Odd\n", num);**

**}**

**printf("\n Enter month number (1-12): ");**

**scanf("%d", &month);**

**switch (month) {**

**case 1: printf("January\n"); break;**

**case 2: printf("February\n"); break;**

**case 3: printf("March\n"); break;**

**case 4: printf("April\n"); break;**

**case 5: printf("May\n"); break;**

**case 6: printf("June\n"); break;**

**case 7: printf("July\n"); break;**

**case 8: printf("August\n"); break;**

**case 9: printf("September\n"); break;**

**case 10: printf("October\n"); break;**

**case 11: printf("November\n"); break;**

**case 12: printf("December\n"); break;**

**default:**

**printf("Invalid month number\n");**

**}**

**return 0;**

**}**

**Output:-**

**Enter a number: 12**

**12 is Even**

**Enter month number (1-12): 10**

**October**

**6. Looping in C:-**

**THEORY EXERCISE: Compare and contrast while loops, for loops, and do-while loops. Explain the scenarios in which each loop is most appropriate.**

**Answer:**

## **1. while Loop:-**

* **Condition is checked first, then the loop executes.**
* **If the condition is false initially, the loop body may never run.**
* **Best for situations where the number of iterations is not known.**

### **Example:-**

**while (i < 5) {**

**printf("%d ", i);**

**i++;**

**}**

### **Use When:-**

* **You don’t know how many times the loop should repeat.**
* **Example: reading input until the user enters 0.**

## **2. for Loop:-**

* **Used when the number of iterations is known or fixed.**
* **All parts (initialization, condition, increment) are written in one line.**
* **Makes the loop clean and easy to understand.**

### 

### 

### **Example:-**

**for (i = 1; i <= 5; i++) {**

**printf("%d ", i);**

**}**

### **Use When:-**

* **loop runs a specific number of times.**
* **Example: printing 1 to 100, iterating through arrays.**

## **3. do-while Loop:-**

* **Loop runs at least one time, even if the condition is false.**
* **Condition is checked after executing the loop body.**

### **Example:**

**do {**

**printf("%d ", i);**

**i++;**

**} while (i < 5);**

### **Use When:**

* **You want the loop body to run once before checking-condition.**
* **Example: menu systems, user input validation.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Loop Type** | **Condition Check** | **Runs at Least Once?** | **Best Use Case** |
| **while** | **Before loop** | **❌ No** | **Unknown number of iterations** |
| **for** | **Before loop** | **❌ No** | **Fixed number of iterations** |
| **do-while** | **After loop** | **✅ Yes** | **User input / menu tasks** |

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

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int i;**

**i = 1;**

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

**while (i <= 10) {**

**printf("%d ", i);**

**i++;**

**}**

**printf("\n\nUsing for loop:\n");**

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

**printf("%d ", i);**

**}**

**i = 1;**

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

**do {**

**printf("%d ", i);**

**i++;**

**} while (i <= 10);**

**return 0;**

**}**

**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**

**7. Loop Control Statements:-**

**THEORY EXERCISE: Explain the use of break, continue, and goto statements in C. Provide examples of each.**

**Answer:-**

# **1. break Statement:-**

### **Use:**

* **The break statement is used to stop a loop or exit a switch case immediately.**
* **When break executes, the control jumps out of the loop/switch.**

### **Example:-**

**for (int i = 1; i <= 5; i++) {**

**if (i == 3) {**

**break;**

**}**

**printf("%d ", i);**

**}**

# **2. continue Statement:-**

### **Use:-**

* **continue skips the current iteration of the loop.**
* **The loop does not stop, it simply jumps to the next iteration.**

### 

### **Example:**

**for (int i = 1; i <= 5; i++) {**

**if (i == 3) {**

**continue; // skips printing 3**

**}**

**printf("%d ", i);**

**}**

# **3. goto Statement:-**

### **Use:**

* **goto is used to jump to a labeled part of the program.**
* **It should be used carefully because it can make programs harder to read.**

### **Example:**

**#include <stdio.h>**

**int main() {**

**goto start;**

**printf("This will not print.\n");**

**start:**

**printf("Jumped using goto.\n");**

**return 0;**

**}**

**LAB EXERCISE: 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.**

**Answer:**

**#include <stdio.h>**

**int main() {**

**int i;**

**printf("Break Example:\n");**

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

**if (i == 5) {**

**break;**

**}**

**printf("%d ", i);**

**}**

**printf("\n\nContinue Example:\n");**

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

**if (i == 3) {**

**continue;**

**}**

**printf("%d ", i);**

**}**

**return 0;**

**}**

**Output:-**

**Break Example:**

**1 2 3 4**

**Continue Example:**

**1 2 4 5 6 7 8 9 10**

**8. Functions in C:-**

**THEORY EXERCISE: What are functions in C? Explain function declaration, definition, and how to call a function. Provide examples.**

**Answer:-Functions are blocks of code that perform a specific task.  
 They help make a program modular, reusable, easy to understand, and easy to debug.**

**C programs are made of many functions, and the most important one is main().**

# **1. Function Declaration (Function Prototype):-**

**This tells the compiler:**

* **The name of the function**
* **The return type**
* **The parameters**

**It does not contain the actual code.**

### **Syntax:**

**return\_type function\_name(parameters)**

**Example:**

**int add(int x, int y);**

# **2. Function Definition:-**

**This contains the actual code that defines what the function does.**

### 

### **Syntax:**

**return\_type function\_name(parameters) {**

**// statements**

**}**

### **Example:**

**int add(int x, int y) {**

**return x + y;**

**}**

# **3. Function Call:-**

**You “call” the function inside main() to actually use it.  
 The program jumps to the function, executes it, and returns the result.**

### **Example:**

**int result = add(5, 3);**

**Example:-**

**#include <stdio.h>**

**// Function Declaration**

**int add(int a, int b);**

**// Function Definition**

**int add(int a, int b) {**

**return a + b;**

**}**

**int main() {**

**int sum;**

**// Function**

**sum = add(10, 20);**

**printf("Sum = %d", sum);**

**return 0;**

**}**

**Output:-**

**Sum = 30**

**LAB EXERCISE: Write a C program that calculates the factorial of a number using a function. Include function declaration, definition, and call.**

**Answer:**

**#include <stdio.h>**

**// Function Declaration**

**int factorial(int n);**

**// Function Definition**

**int factorial(int n) {**

**int i, fact = 1;**

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

**fact = fact \* i;**

**}**

**return fact;**

**}**

**int main() {**

**int num;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**// Function Call**

**printf("Factorial of %d = %d", num, factorial(num));**

**return 0;**

**}**

**Output:-**

**Enter a number: 5**

**Factorial of 5 = 120**

**9. Arrays in C:-**

**THEORY EXERCISE: Explain the concept of arrays in C. Differentiate between one-dimensional and multi-dimensional arrays with examples.**

**Answer:An array is a collection of elements of the same data type, stored in contiguous memory locations.  
 It allows you to store many values under a single variable name by using index numbers.**

**Example:**

**int marks[5]; // array of 5 integers**

# **1. One-Dimensional Array (1D Array):-**

* **A 1D array stores values in a single row.It is like a simple list.**

### **Example:**

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

### **Accessing elements:-**

**printf("%d", numbers[2]); // prints 30**

**Use Case:-Storing marks of students, list of prices, etc.**

# **2. Multi-Dimensional Array (2D Array):-**

* **A multi-dimensional array stores data in rows and columns (like a table).**

**The most common type is a 2D array.**

### **Example (2D array):**

**int matrix[2][3] = {{1, 2, 3},{4, 5, 6}};**

**This means:**

* **2 rows**
* **3 columns**

### **Accessing elements:**

**printf("%d", matrix[1][2]); // prints 6**

### **Use Case:Storing matrices, tables, game boards, etc.**

# **Difference Between 1D and Multi-Dimensional Arrays:-**

# 

|  |  |  |
| --- | --- | --- |
| **Feature** | **1D Array** | **Multi-Dimensional Array** |
| Structure | Single row (list) | Rows and columns (table) |
| Syntax | int a[5]; | int a[3][4]; |
| Storage | Linear | Grid-like |
| Access | a[i] | a[row][column] |
| Example | Marks list | Matrix / table |

# 

**LAB EXERCISE: 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.**

**Answer:**

**#include <stdio.h>**

**int main() {**

**int arr[5];**

**int matrix[3][3];**

**int i, j, sum = 0;**

**// -------- 1D Array Input --------**

**printf("Enter 5 integers:\n");**

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

**scanf("%d", &arr[i]);**

**}**

**// -------- Print 1D Array --------**

**printf("\nYou entered (1D Array): ");**

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

**printf("%d ", arr[i]);**

**}**

**// -------- 2D Array Input --------**

**printf("\n\nEnter elements of 3x3 matrix:\n");**

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

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

**scanf("%d", &matrix[i][j]);**

**sum += matrix[i][j];**

**}**

**}**

**// -------- Print 2D Array --------**

**printf("\nThe 3x3 Matrix is:\n");**

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

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

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

**}**

**printf("\n");**

**}**

**// -------- Sum of Matrix --------**

**printf("\nSum of all matrix elements = %d\n", sum);**

**return 0;**

**}**

**Output:-**

**Enter 5 integers:**

**10**

**20**

**30**

**40**

**50**

**You entered (1D Array): 10 20 30 40 50**

**Enter elements of 3x3 matrix:**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**The 3x3 Matrix is:**

**1 2 3**

**4 5 6**

**7 8 9**

**Sum of all matrix elements = 45**

**10. Pointers in C:-**

**THEORY EXERCISE: Explain what pointers are in C and how they are declared and initialized. Why are pointers important in C?**

**Answer:A pointer is a special variable that stores the memory address of another variable.  
 Instead of holding a value directly, a pointer holds the location where the value is stored.**

**Example:  
 If int a = 10; is stored at memory address 1000,  
 a pointer can store 1000.**

**Pointer Declaration:-**

**You declare a pointer using the \* symbol.**

### **Syntax**

**data\_type \*pointer\_name;**

### **Example**

**int \*p; // p can store address of an int**

# **Pointer Initialization:-**

**You assign the address of a variable to the pointer using the address operator (&).**

### **Example**

**int a = 10;**

**int \*p = &a; // p stores the address of a**

**Here:**

* **a → value = 10**
* **&a → address of a**
* **p → stores this address**
* **\*p → gives the value stored at that address (10)**

# **Why Are Pointers Important in C?**

**Pointers are extremely important because they allow you to:**

### **1. Access memory directly**

**C gives low-level memory control through pointers.**

### **2. Work with arrays and strings efficiently**

**Arrays, strings, and pointer arithmetic are closely related.**

### **3. Pass large data to functions**

**Using pointers avoids copying large data — improves performance.**

### **4. Create dynamic memory**

**Functions like malloc() and free() require pointers.**

### **5. Handle complex data structures**

**Pointers are used to build:**

* **Linked lists**
* **Trees**
* **Graphs**
* **Queues**
* **Stacks**

**LAB EXERCISE: Write a C program to demonstrate pointer usage. Use a pointer to modify the value of a variable and print the result.**

**Answer:**

**#include <stdio.h>**

**int main() {**

**int a = 10;**

**int \*p;**

**p = &a;**

**printf("Original value of a: %d\n", a);**

**\*p = 20;**

**printf("Modified value of a using pointer: %d\n", a);**

**return 0;**

**}**

**output:-**

**Original value of a: 10**

**Modified value of a using pointer: 20**

**11. Strings in C:-**

**THEORY EXERCISE: Explain string handling functions like strlen(), strcpy(), strcat(), strcmp(), and strchr(). Provide examples of when these functions are useful.**

**Answer:**

**1. strlen():-**

### **Meaning:**

**Returns the length of a string (number of characters, excluding '\0').**

### **Syntax:**

**strlen(string);**

### **Example:**

**strlen("Hello"); // returns 5**

### **Use:**

**To find how long a string is (useful in input validation, loops, etc.).**

# **2. strcpy():-**

### **Meaning:**

**Copies one string into another.**

### **Syntax:**

**strcpy(destination, source);**

### **Example:**

**char a[20], b[] = "World";**

**strcpy(a, b); // a becomes "World"**

### **Use:**

**Used when you want to copy names, messages, or input strings.**

# **3. strcat():-**

### **Meaning:**

**Joins (concatenates) two strings.**

### **Syntax:**

**strcat(string1, string2);**

### **Example:**

**char a[30] = "Hello";**

**char b[] = "World";**

**strcat(a, b); // a becomes "HelloWorld"**

### **Use:**

**Useful for merging two strings like first name + last name.**

**4. strcmp():-**

### **Meaning:Compares two strings.**

### **Returns:**

* **0 → if both strings are equal**
* **> 0 → if first string is greater**
* **< 0 → if second string is greater**

### **Example:**

**strcmp("apple", "apple"); // returns 0**

**strcmp("cat", "bat"); // returns >0**

### **Use:**

**Used in sorting names, checking passwords, matching input, etc.**

# **5. strchr():-**

### **Meaning:**

**Finds the first occurrence of a character in a string.**

### **Syntax:**

**strchr(string, character);**

### **Example:**

**strchr("Hello", 'l'); // returns pointer to first 'l'**

**LAB EXERCISE: 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().**

**Answer:**

**#include <stdio.h>**

**#include <string.h>**

**int main() {**

**char str1[100], str2[100];**

**printf("Enter first string: ");**

**gets(str1); // you can also use fgets()**

**printf("Enter second string: ");**

**gets(str2);**

**strcat(str1, str2);**

**printf("\nConcatenated String: %s", str1);**

**printf("\nLength of Concatenated String: %lu", strlen(str1));**

**return 0;**

**}**

**output:-**

**Enter first string: Hemanshu**

**Enter second string: Pawar**

**Concatenated String: HemanshuPawar**

**Length of Concatenated String: 13**

**12. Structures in C:-**

**THEORY EXERCISE: Explain the concept of structures in C. Describe how to declare, initialize, and access structure members.**

**Answer:-**

**A structure is a user-defined data type in C that allows you to group different types of data under one name.**

**For example, a student may have:**

* **name (string)**
* **roll number (int)**
* **marks (float)**

**A structure can hold all of these together.**

# **1. Declaring a Structure:-**

**You use the keyword struct.**

### **Syntax:**

**struct structure\_name {**

**data\_type member1;**

**data\_type member2;**

**...**

**};**

### **Example:**

**struct Student {**

**int roll;**

**float marks;**

**char name[20];**

**};**

# **2. Declaring Structure Variables:-**

### **Example:**

**struct Student s1;**

# **3. Initializing Structure Members:-**

**You can assign values in two ways.**

### **Method 1: Direct initialization**

**struct Student s1 = {1, 95.5, "Rahul"};**

### **Method 2: Assign values separately**

**struct Student s1;**

**s1.roll = 1;**

**s1.marks = 95.5;**

**strcpy(s1.name, "Rahul"); // for strings**

# **4. Accessing Structure Members:-**

**Use the dot (.) operator.**

### **Example:**

**printf("%d", s1.roll);**

**printf("%f", s1.marks);**

**printf("%s", s1.name);**

# 

# 

# **Example:-**

**#include <stdio.h>**

**#include <string.h>**

**struct Student {**

**int roll;**

**float marks;**

**char name[20];**

**};**

**int main() {**

**struct Student s1;**

**s1.roll = 101;**

**s1.marks = 88.5;**

**strcpy(s1.name, "hemanshu");**

**printf("Name: %s\n", s1.name);**

**printf("Roll: %d\n", s1.roll);**

**printf("Marks: %.1f\n", s1.marks);**

**return 0;**

**}**

**Output:-**

**Name: hemanshu**

**Roll: 101**

**Marks: 88.5**

**LAB EXERCISE: o Write a C program that defines a structure to store a student's details (name, roll number, and marks). Use an array of structures to store details of 3 students and print them.**

**Answer:-**

**#include <stdio.h>**

**struct Student {**

**char name[50];**

**int roll;**

**float marks;**

**};**

**int main() {**

**struct Student s[3];**

**int i;**

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

**printf("\nEnter details of student %d:\n", i + 1);**

**printf("Enter Name: ");**

**scanf("%s", s[i].name);**

**printf("Enter Roll Number: ");**

**scanf("%d", &s[i].roll);**

**printf("Enter Marks: ");**

**scanf("%f", &s[i].marks);**

**}**

**printf("\n--- Student Details ---\n");**

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

**printf("\nStudent %d:\n", i + 1);**

**printf("Name: %s\n", s[i].name);**

**printf("Roll Number: %d\n", s[i].roll);**

**printf("Marks: %.2f\n", s[i].marks);**

**}**

**return 0;**

**}**

**Output:-**

**Enter details of student 1:**

**Enter Name: hemanshu**

**Enter Roll Number: 101**

**Enter Marks: 89.5**

**Enter details of student 2:**

**Enter Name: vinay**

**Enter Roll Number: 102**

**Enter Marks: 92.0**

**Enter details of student 3:**

**Enter Name: abhay**

**Enter Roll Number: 103**

**Enter Marks: 85.75**

**--- Student Details ---**

**Student 1:**

**Name: hemanshu**

**Roll Number: 101**

**Marks: 89.50**

**Student 2:**

**Name: vinay**

**Roll Number: 102**

**Marks: 92.00**

**Student 3:**

**Name: abhay**

**Roll Number: 103**

**Marks: 85.75**

**13. File Handling in C:-**

**THEORY EXERCISE: Explain the importance of file handling in C. Discuss how to perform file operations like opening, closing, reading, and writing files.**

**Answer:**

**File handling in C allows a program to store data permanently, read data from files, and manage information efficiently.  
 Without file handling, all program data would be lost once the program ends because variables store information only in RAM (temporary).**

**File handling is essential for applications like:**

* **Storing student records**
* **Saving logs**
* **Reading configuration files**
* **Writing reports**
* **Processing large data sets**

**Importance of file handling:-**

* **Permanent Storage**
* **Data Sharing**
* **Large Data Handling**
* **Backup & Security**
* **Automation**

**File operations:-**

## **1. Opening a File:-**

**Files are opened using the fopen() function.**

### **Syntax:-**

### **FILE \*ptr;**

### **ptr = fopen("filename.txt", "mode");**

## **2. Closing a File:-**

**After operations, the file must be closed.**

### **Syntax:-**

**fclose(ptr);**

**Closing prevents:**

* **Memory leaks**
* **File corruption**
* **Data loss**

## **3. Writing to a File:-**

**To write data, we use:**

* **fprintf() — writes formatted text**
* **fputs() — writes simple strings**
* **fputc() — writes a single character**

### **Example:-**

### **fprintf(ptr, "Hello World");**

## **4. Reading from a File:-**

**To read data from a file, we use:**

* **fscanf() — formatted input (like scanf)**
* **fgets() — reads a full line**
* **fgetc() — reads a single character**

### **Example**

**fgets(str, 50, ptr);**

**LAB EXERCISE: Write a C program to create a file, write a string into it, close the file, then open the file again to read and display its contents.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**FILE \*fp;**

**char text[100];**

**// ----- Writing to the file -----**

**fp = fopen("myfile.txt", "w");**

**if (fp == NULL) {**

**printf("Error opening file!\n");**

**return 1;**

**}**

**printf("Enter a line of text: ");**

**fgets(text, sizeof(text), stdin);**

**fputs(text, fp);**

**fclose(fp);**

**// ----- Reading from the file -----**

**fp = fopen("myfile.txt", "r");**

**if (fp == NULL) {**

**printf("Error opening file!\n");**

**return 1;**

**}**

**printf("\nContent of the file:\n");**

**while (fgets(text, sizeof(text), fp) != NULL) {**

**printf("%s", text);**

**}**

**fclose(fp);**

**return 0;**

**}**

**Output:-**

**Enter a line of text: Hello, this is a test file.**

**Content of the file:**

**Hello, this is a test file.**

**EXTRA LAB EXERCISES FOR IMPROVING PROGRAMMING LOGIC:-**

**1. Operators:-**

**LAB EXERCISE 1: Simple Calculator**

* **Write a C program that acts as a simple calculator. The program should take two numbers and an operator as input from the user and perform the respective operation (addition, subtraction, multiplication, division, or modulus) using operators. Challenge: Extend the program to handle invalid operator inputs.**

**Answer:**

**#include <stdio.h>**

**int main() {**

**int a, b;**

**char op;**

**printf("Enter two numbers: ");**

**scanf("%d %d", &a, &b);**

**printf("Enter an operator (+, -, \*, /, %%) : ");**

**scanf(" %c", &op);**

**switch(op) {**

**case '+':**

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

**break;**

**case '-':**

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

**break;**

**case '\*':**

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

**break;**

**case '/':**

**if (b != 0)**

**printf("Result: %d / %d = %.2f\n", a, b, (float)a / b);**

**else**

**printf("Error: Division by zero is not allowed!\n");**

**break;**

**case '%':**

**if (b != 0)**

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

**else**

**printf("Error: Modulus by zero is not allowed!\n");**

**break;**

**default:**

**printf("Error: Invalid operator '%c'\n", op);**

**}**

**return 0;**

**}**

**Output:-**

**Enter two numbers: 15 3**

**Enter an operator (+, -, \*, /, %): /**

**Result: 15 / 3 = 5.00**

**LAB EXERCISE 2: Check Number Properties Write a C program that takes an integer from the user and checks the following using different operators: o Whether the number is even or odd. o Whether the number is positive, negative, or zero. o Whether the number is a multiple of both 3 and 5.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int num;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**if (num % 2 == 0)**

**printf("The number is Even.\n");**

**else**

**printf("The number is Odd.\n");**

**// Check positive, negative or zero**

**if (num > 0)**

**printf("The number is Positive.\n");**

**else if (num < 0)**

**printf("The number is Negative.\n");**

**else**

**printf("The number is Zero.\n");**

**// Check multiple of both 3 and 5**

**if (num % 3 == 0 && num % 5 == 0)**

**printf("The number is a Multiple of both 3 and 5.\n");**

**else**

**printf("The number is not a multiple of both 3 and 5.\n");**

**return 0;**

**}**

**Output:-**

**Enter a number: 15**

**The number is Odd.**

**The number is Positive.**

**The number is a Multiple of both 3 and 5.**

# **2. Control Statements:-**

## **LAB EXERCISE 1: Grade Calculator**

* **Write a C program that takes the marks of a student as input and displays the corresponding grade based on the following conditions:**

**o Marks > 90: Grade A**

**o Marks > 75 and <= 90: Grade B**

**o Marks > 50 and <= 75: Grade C**

**o Marks <= 50: Grade D**

**Use if-else or switch statements for the decision-making process.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int marks;**

**printf("Enter student's marks: ");**

**scanf("%d", &marks);**

**if (marks > 90) {**

**printf("Grade: A\n");**

**}**

**else if (marks > 75 && marks <= 90) {**

**printf("Grade: B\n");**

**}**

**else if (marks > 50 && marks <= 75) {**

**printf("Grade: C\n");**

**}**

**else {**

**printf("Grade: D\n");**

**}**

**return 0;**

**}**

**Output:**

**Enter student's marks: 82**

**Grade: B**

## **LAB EXERCISE 2: Number Comparison**

* **Write a C program that takes three numbers from the user and determines: o The largest number.**

**o The smallest number.**

* **Challenge: Solve the problem using both if-else and switch-case statements.**

**Answer:**

**#include <stdio.h>**

**int main() {**

**int a, b, c;**

**int largest, smallest;**

**int choice;**

**printf("Enter three numbers: ");**

**scanf("%d %d %d", &a, &b, &c);**

**// Using if-else**

**if (a > b && a > c)**

**largest = a;**

**else if (b > a && b > c)**

**largest = b;**

**else**

**largest = c;**

**if (a < b && a < c)**

**smallest = a;**

**else if (b < a && b < c)**

**smallest = b;**

**else**

**smallest = c;**

**printf("\nLargest (if-else): %d", largest);**

**printf("\nSmallest (if-else): %d", smallest);**

**printf("\n\nEnter 1 to show result using switch: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1:**

**printf("\nLargest (switch): %d", largest);**

**printf("\nSmallest (switch): %d", smallest);**

**break;**

**default:**

**printf("Invalid choice!");**

**}**

**return 0;**

**}**

**Output:-**

**Enter three numbers: 12 5 20**

**Largest (if-else): 20**

**Smallest (if-else): 5**

**Enter 1 to show result using switch: 1**

**Largest (switch): 20**

**Smallest (switch): 5**

# **3. Loops:-**

## **LAB EXERCISE 1: Prime Number Check**

* **Write a C program that checks whether a given number is a prime number or not using a for loop.**
* **Challenge: Modify the program to print all prime numbers between 1 and a given number.**

**Answer:**

**#include <stdio.h>**

**int main() {**

**int n, i, count = 0;**

**printf("Enter a number: ");**

**scanf("%d", &n);**

**// --------- Check if the number is prime ---------**

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

**if (n % i == 0)**

**count++;**

**}**

**if (count == 2)**

**printf("%d is a Prime number.\n", n);**

**else**

**printf("%d is NOT a Prime number.\n", n);**

**// --------- all prime numbers from 1 to n ---------**

**printf("\nPrime numbers from 1 to %d:\n", n);**

**for (int num = 2; num <= n; num++) {**

**int c = 0;**

**for (i = 1; i <= num; i++) {**

**if (num % i == 0)**

**c++;**

**}**

**if (c == 2)**

**printf("%d ", num);**

**}**

**return 0;**

**}**

**Output:-**

**Enter a number: 15**

**15 is NOT a Prime number.**

**Prime numbers from 1 to 15:**

**2 3 5 7 11 13**

# 

## **LAB EXERCISE 2: Multiplication Table**

* **Write a C program that takes an integer input from the user and prints its multiplication table using a for loop. Challenge: Allow the user to input the range of the multiplication table (e.g., from 1 to N).**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int num, start, end, i;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**printf("Enter starting range: ");**

**scanf("%d", &start);**

**printf("Enter ending range: ");**

**scanf("%d", &end);**

**printf("\nMultiplication Table of %d from %d to %d:\n\n", num, start, end);**

**for (i = start; i <= end; i++) {**

**printf("%d x %d = %d\n", num, i, num \* i);**

**}**

**return 0;**

**}**

**Output:-**

**Enter a number: 7**

**Enter starting range: 1**

**Enter ending range: 10**

**Multiplication Table of 7 from 1 to 10:**

**7 x 1 = 7**

**7 x 2 = 14**

**7 x 3 = 21**

**7 x 4 = 28**

**7 x 5 = 35**

**7 x 6 = 42**

**7 x 7 = 49**

**7 x 8 = 56**

**7 x 9 = 63**

**7 x 10 = 70**

## **LAB EXERCISE 3: Sum of Digits**

* **Write a C program that takes an integer from the user and calculates the sum of its digits using a while loop.**
* **Challenge: Extend the program to reverse the digits of the number.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int num, temp, digit, sum = 0, reverse = 0;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**temp = num;**

**while (temp > 0) {**

**digit = temp % 10;**

**sum = sum + digit;**

**reverse = reverse \* 10 + digit;**

**temp = temp / 10;**

**}**

**printf("Sum of digits = %d\n", sum);**

**printf("Reverse of the number = %d\n", reverse);**

**return 0;**

**}**

**Output:-**

**Enter a number: 4567**

**Sum of digits = 22**

**Reverse of the number = 7654**

# **4. Arrays:-**

## **LAB EXERCISE 1: Maximum and Minimum in Array**

* **Write a C program that accepts 10 integers from the user and stores them in an array. The program should then find and print the maximum and minimum values in the array.**
* **Challenge: Extend the program to sort the array in ascending order.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int a[10], i, j, temp;**

**int max, min;**

**printf("Enter 10 integers:\n");**

**for (i = 0; i < 10; i++) {**

**scanf("%d", &a[i]);**

**}**

**max = min = a[0];**

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

**if (a[i] > max)**

**max = a[i];**

**if (a[i] < min)**

**min = a[i];**

**}**

**// sorting (ascending)**

**for (i = 0; i < 10; i++) {**

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

**if (a[i] > a[j]) {**

**temp = a[i];**

**a[i] = a[j];**

**a[j] = temp;**

**}**

**}**

**}**

**printf("\nMaximum value = %d", max);**

**printf("\nMinimum value = %d", min);**

**printf("\n\nArray in ascending order:\n");**

**for (i = 0; i < 10; i++) {**

**printf("%d ", a[i]);**

**}**

**return 0;**

**}**

**Output:-**

**Enter 10 integers:**

**5 8 1 9 2 7 6 3 4 10**

**Maximum value = 10**

**Minimum value = 1**

**Array in ascending order:**

**1 2 3 4 5 6 7 8 9 10**

## **LAB EXERCISE 2: Matrix Addition**

* **Write a C program that accepts two 2x2 matrices from the user and adds them. Display the resultant matrix.**
* **Challenge: Extend the program to work with 3x3 matrices and matrix multiplication.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int a[3][3], b[3][3], sum[3][3], mul[3][3];**

**int i, j, k;**

**printf("Enter elements of 3x3 Matrix A:\n");**

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

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

**scanf("%d", &a[i][j]);**

**}**

**}**

**printf("Enter elements of 3x3 Matrix B:\n");**

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

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

**scanf("%d", &b[i][j]);**

**}**

**}**

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

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

**sum[i][j] = a[i][j] + b[i][j];**

**}**

**}**

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

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

**mul[i][j] = 0;**

**for (k = 0; k < 3; k++) {**

**mul[i][j] += a[i][j] \* b[j][k];**

**}**

**}**

**}**

**printf("\nSum of the two matrices:\n");**

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

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

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

**}**

**printf("\n");**

**}**

**printf("\nMultiplication of the two matrices:\n");**

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

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

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

**}**

**printf("\n");**

**}**

**return 0;**

**}**

**Output:-**

**Enter elements of 3x3 Matrix A:**

**1 2 3**

**4 5 6**

**7 8 9**

**Enter elements of 3x3 Matrix B:**

**9 8 7**

**6 5 4**

**3 2 1**

**Sum of the two matrices:**

**10 10 10**

**10 10 10**

**10 10 10**

**Multiplication of the two matrices:**

**30 24 18**

**84 69 54**

**138 114 90**

## **LAB EXERCISE 3: Sum of Array Elements**

* **Write a C program that takes N numbers from the user and stores them in an array. The program should then calculate and display the sum of all array elements.**
* **Challenge: Modify the program to also find the average of the numbers.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int n, i;**

**float sum = 0, avg;**

**printf("Enter how many numbers: ");**

**scanf("%d", &n);**

**int arr[n];**

**printf("Enter %d numbers:\n", n);**

**for (i = 0; i < n; i++) {**

**scanf("%d", &arr[i]);**

**sum += arr[i];**

**}**

**avg = sum / n;**

**printf("\nSum = %.2f", sum);**

**printf("\nAverage = %.2f", avg);**

**return 0;**

**}**

**Output:-**

**Enter how many numbers: 5**

**Enter 5 numbers:**

**10**

**20**

**30**

**40**

**50**

**Sum = 150.00**

**Average = 30.00**

# **5. Functions:-**

## **LAB EXERCISE 1: Fibonacci Sequence**

**Write a C program that generates the Fibonacci sequence up to N terms using a recursive function.**

**Challenge: Modify the program to calculate the Nth Fibonacci number using both iterative and recursive methods. Compare their efficiency.**

**Answer:-**

**#include <stdio.h>**

**int fib\_rec(int n) {**

**if (n == 0) return 0;**

**if (n == 1) return 1;**

**return fib\_rec(n - 1) + fib\_rec(n - 2);**

**}**

**int fib\_iter(int n) {**

**int a = 0, b = 1, next, i;**

**if (n == 0) return 0;**

**for (i = 2; i <= n; i++) {**

**next = a + b;**

**a = b;**

**b = next;**

**}**

**return b;**

**}**

**int main() {**

**int N, i;**

**printf("Enter N: ");**

**scanf("%d", &N);**

**printf("\nFibonacci sequence (recursive):\n");**

**for (i = 0; i < N; i++) {**

**printf("%d ", fib\_rec(i));**

**}**

**printf("\nNth Fibonacci (recursive) = %d\n", fib\_rec(N));**

**printf("\nNth Fibonacci (iterative) = %d\n", fib\_iter(N));**

**return 0;**

**}**

**Output:-**

**Enter N: 7**

**Fibonacci sequence (recursive):**

**0 1 1 2 3 5 8**

**Nth Fibonacci (recursive) = 13**

**Nth Fibonacci (iterative) = 13**

## **LAB EXERCISE 2: Factorial Calculation**

* **Write a C program that calculates the factorial of a given number using a function.**
* **Challenge: Implement both an iterative and a recursive version of the factorial function and compare their performance for large numbers.**

**Answer:-**

**#include <stdio.h>**

**int factorial(int n) {**

**int i, fact = 1;**

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

**fact \*= i;**

**}**

**return fact;**

**}**

**int main() {**

**int n;**

**printf("Enter a number: ");**

**scanf("%d", &n);**

**if (n < 0)**

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

**else**

**printf("Factorial of %d = %d\n", n, factorial(n));**

**return 0;**

**}**

**Output:-**

**Enter a number: 5**

**Factorial of 5 = 120**

## **LAB EXERCISE 3: Palindrome Check**

* **Write a C program that takes a number as input and checks whether it is a palindrome using a function.**
* **Challenge: Modify the program to check if a given string is a palindrome.**

**Answer:-**

**#include <stdio.h>**

**#include <string.h>**

**int main() {**

**int num, temp, rev = 0, d;**

**char str[50];**

**int i, j;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**temp = num;**

**while (temp > 0) {**

**d = temp % 10;**

**rev = rev \* 10 + d;**

**temp = temp / 10;**

**}**

**if (rev == num)**

**printf("%d is a palindrome number.\n", num);**

**else**

**printf("%d is not a palindrome number.\n", num);**

**printf("\nEnter a string: ");**

**scanf("%s", str);**

**i = 0;**

**j = strlen(str) - 1;**

**while (i < j) {**

**if (str[i] != str[j])**

**break;**

**i++;**

**j--;**

**}**

**if (i >= j)**

**printf("%s is a palindrome string.\n", str);**

**else**

**printf("%s is not a palindrome string.\n", str);**

**return 0;**

**}**

**Output:-**

**Enter a number: 121**

**121 is a palindrome number.**

**Enter a string: hemanshu**

**hemanshu is not palindrome string.**

# **6. Strings:-**

## **LAB EXERCISE 1: String Reversal**

* **Write a C program that takes a string as input and reverses it using a function.**
* **Challenge: Write the program without using built-in string handling functions.**

**Answer:**

**#include <stdio.h>**

**void reverse(char s[]) {**

**int i = 0, j, len = 0;**

**char temp;**

**while (s[len] != '\0') {**

**len++;**

**}**

**j = len - 1;**

**while (i < j) {**

**temp = s[i];**

**s[i] = s[j];**

**s[j] = temp;**

**i++;**

**j--;**

**}**

**}**

**int main() {**

**char str[50];**

**printf("Enter a string: ");**

**scanf("%s", str);**

**reverse(str);**

**printf("Reversed string: %s\n", str);**

**return 0;**

**}**

**Output:-**

**Enter a string: hello**

**Reversed string: olleh**

## **LAB EXERCISE 2: Count Vowels and Consonants**

* **Write a C program that takes a string from the user and counts the number of vowels and consonants in the string.**
* **Challenge: Extend the program to also count digits and special characters.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**char str[100];**

**int i = 0;**

**int vowels = 0, cons = 0, digits = 0, special = 0;**

**printf("Enter a string: ");**

**gets(str);**

**while (str[i] != '\0') {**

**if (str[i] == 'a' || str[i] == 'e' || str[i] == 'i' ||**

**str[i] == 'o' || str[i] == 'u' ||**

**str[i] == 'A' || str[i] == 'E' || str[i] == 'I' ||**

**str[i] == 'O' || str[i] == 'U') {**

**vowels++;**

**}**

**else if ((str[i] >= 'a' && str[i] <= 'z') ||**

**(str[i] >= 'A' && str[i] <= 'Z')) {**

**cons++;**

**}**

**else if (str[i] >= '0' && str[i] <= '9') {**

**digits++;**

**}**

**else {**

**special++;**

**}**

**i++;**

**}**

**printf("\nVowels: %d", vowels);**

**printf("\nConsonants: %d", cons);**

**printf("\nDigits: %d", digits);**

**printf("\nSpecial Characters: %d\n", special);**

**return 0;**

**}**

**Output:-**

**Enter a string: Hello123@#**

**Vowels: 2**

**Consonants: 3**

**Digits: 3**

**Special Characters: 2**

## **LAB EXERCISE 3: Word Count**

* **Write a C program that counts the number of words in a sentence entered by the user.**
* **Challenge: Modify the program to find the longest word in the sentence.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**char str[200];**

**int i = 0, words = 0, len = 0, maxLen = 0;**

**int start = 0, maxStart = 0;**

**printf("Enter a sentence: ");**

**gets(str);**

**while (str[i] != '\0') {**

**if (str[i] != ' ' && str[i] != '\t') {**

**len++;**

**if (len == 1)**

**start = i;**

**}**

**else {**

**if (len > 0) {**

**words++;**

**if (len > maxLen) {**

**maxLen = len;**

**maxStart = start;**

**}**

**len = 0;**

**}**

**}**

**i++;**

**}**

**if (len > 0) {**

**words++;**

**if (len > maxLen) {**

**maxLen = len;**

**maxStart = start;**

**}**

**}**

**printf("\nTotal words: %d\n", words);**

**printf("Longest word: ");**

**for (i = maxStart; i < maxStart + maxLen; i++) {**

**printf("%c", str[i]);**

**}**

**printf("\n");**

**return 0;**

**}**

**output:-**

**Enter a sentence: my name is hemanshu**

**Total words: 4**

**Longest word: hemanshu**

**Extra Logic Building Challenges:-**

# **Lab Challenge 1: Armstrong Number**

* **Write a C program that checks whether a given number is an Armstrong number or not (e.g., 153 = 1^3 + 5^3 + 3^3).**
* **Challenge: Write a program to find all Armstrong numbers between 1 and 1000.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int num, temp, digit, sum = 0;**

**printf("Enter a number: ");**

**scanf("%d", &num);**

**temp = num;**

**sum = 0;**

**while (temp > 0) {**

**digit = temp % 10;**

**sum = sum + (digit \* digit \* digit);**

**temp = temp / 10;**

**}**

**if (sum == num)**

**printf("%d is an Armstrong number.\n", num);**

**else**

**printf("%d is not an Armstrong number.\n", num);**

**/\* Challenge: Armstrong numbers from 1 to 1000 \*/**

**printf("\nArmstrong numbers between 1 and 1000:\n");**

**for (num = 1; num <= 1000; num++) {**

**temp = num;**

**sum = 0;**

**while (temp > 0) {**

**digit = temp % 10;**

**sum = sum + (digit \* digit \* digit);**

**temp = temp / 10;**

**}**

**if (sum == num) {**

**printf("%d ", num);**

**}**

**}**

**printf("\n");**

**return 0;**

**}**

**Output:-**

**Enter a number: 153**

**153 is an Armstrong number.**

**Armstrong numbers between 1 and 1000:**

**1 153 370 371 407**

## **Lab Challenge 2: Pascal’s Triangle**

* **Write a C program that generates Pascal’s Triangle up to N rows using loops.**
* **Challenge: Implement the same program using a recursive function.**

**Answer:-**

**#include <stdio.h>**

**int main() {**

**int n, i, j, num;**

**printf("Enter number of rows: ");**

**scanf("%d", &n);**

**for (i = 0; i < n; i++) {**

**num = 1;**

**for (j = 0; j < n - i - 1; j++) {**

**printf(" ");**

**}**

**for (j = 0; j <= i; j++) {**

**printf("%d ", num);**

**num = num \* (i - j) / (j + 1);**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

**Output:-**

**Enter number of rows: 5**

**1**

**1 1**

**1 2 1**

**1 3 3 1**

**1 4 6 4 1**

## **Lab Challenge 3: Number Guessing Game**

* **Write a C program that implements a simple number guessing game. The program should generate a random number between 1 and 100, and the user should guess the number within a limited number of attempts.**
* **Challenge: Provide hints to the user if the guessed number is too high or too low.**

**Answer:-**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**int main() {**

**int num, guess, tries = 0;**

**srand(time(0));**

**num = rand() % 100 + 1;**

**printf("Guess the number (1 to 100):\n");**

**while (tries < 7) { // limited attempts**

**printf("Enter your guess: ");**

**scanf("%d", &guess);**

**tries++;**

**if (guess == num) {**

**printf("Congratulations! You guessed it in %d attempts.\n", tries);**

**break;**

**}**

**else if (guess > num) {**

**printf("Too high! Try again.\n");**

**}**

**else {**

**printf("Too low! Try again.\n");**

**}**

**}**

**if (tries == 7 && guess != num) {**

**printf("\nOut of attempts! The correct number was %d.\n", num);**

**}**

**return 0;**

**}**

**Answer:-**

**Guess the number (1 to 100):**

**Enter your guess: 50**

**Too high! Try again.**

**Enter your guess: 20**

**Too low! Try again.**

**Enter your guess: 35**

**Congratulations! You guessed it in 3 attempts.**

## 