Lenora college of engineering – Rampachodavaram

CSE – B.Tech – C Language Lab - R23

UNIT I

WEEK 1

Objective: Getting familiar with the programming environment on the computer and writing the first program.

Suggested Experiments/Activities:

Tutorial 1: Problem-solving using Computers.

Lab1: Familiarization with programming environment

i) Basic Linux environment and its editors like Vi, Vim & Emacs etc.
ii) Exposure to Turbo C, gcc
iii) Writing simple programs using printf(), scanf()

WEEK 2

Objective: Getting familiar with how to formally describe a solution to a problem in a series of finite steps both using textual notation and graphic notation.

Suggested Experiments /Activities:

Tutorial 2: Problem-solving using Algorithms and Flow charts.

Lab 1: Converting algorithms/flow charts into C Source code.

Developing the algorithms/flowcharts for the following sample programs

i) Sum and average of 3 numbers
ii) Conversion of Fahrenheit to Celsius and vice versa
iii) Simple interest calculation

WEEK 3

Objective: Learn how to define variables with the desired data-type, initialize them with appropriate values and how arithmetic operators can be used with variables and constants.

Suggested Experiments/Activities:

Tutorial 3: Variable types and type conversions:

Lab 3: Simple computational problems using arithmetic expressions.

i) Finding the square root of a given number
ii) Finding compound interest
iii) Area of a triangle using heron's formulae
iv) Distance travelled by an object

Unit 1

Week 1

Tutorial – 1

Lab 1

i) Basic Linux environment and its editors like Vi, Vim & Emacs etc.

Basic Linux Environment and Its Editors (vi, vim, emacs) – In Simple Terms

1. What is Linux Environment?

Linux is an operating system widely used in servers, programming, cybersecurity, networking, and embedded systems.

In a Linux environment, you interact mainly through:

a) Shell / Terminal

- It is a command-line interface.
- You type commands here to work with files, directories, programs, etc.

Common shells:

- **bash** (most widely used)
- **zsh**
- **sh**

b) File System Structure

Linux follows a tree-like format starting with / (root).

Important directories:

- **/home** – user files
- **/etc** – configuration files
- **/bin** – basic commands
- **/usr** – applications
- **/var** – logs

c) Basic Linux Commands

Purpose	**Command**
List files	ls
Change directory	cd foldername
Print working directory	pwd

Purpose	Command
Create file	touch file.c
Create folder	mkdir test
Copy	cp a b
Move/Rename	mv a b
Remove	rm file
Compile C program	gcc file.c -o output
Run compiled program	./output

---

2. Linux Text Editors

Text editors in Linux help you write code, scripts, configuration files, and programs.

Most used editors:

- **vi**
- **vim**
- **emacs**
- **nano**

---

3. vi Editor

vi is one of the oldest editors in Linux.

Modes in vi

1. **Command Mode**
   - Default mode.
   - You give commands like delete, copy, save, quit.
2. **Insert Mode**
   - You type text.
   - Enter by pressing i.
3. **Last Line Mode**
   - For saving and quitting commands.
   - Enter by pressing :.

Basic Commands

Task	Command
Open file	vi file.c
Insert mode	i
Save	:w
Quit	:q
Save & quit	:wq
Quit without saving	:q!
Delete line	dd
Copy line	yy
Paste	p

---

4. Vim Editor

vim = Vi Improved

It has:

- Syntax highlighting
- Better navigation
- Undo/Redo support
- Plugins

Open file

vim program.c

Extra features vs vi

- Undo: u
- Redo: Ctrl + r
- Highlight text based on language
- Auto-indent

---

5. Emacs Editor

Emacs is a powerful and highly customizable editor.
It is used for writing code, editing files, running terminal commands, even browsing the internet inside the editor.

Open file

emacs file.c

Basic Emacs commands

(Here Ctrl is written as C and Alt is written as M)

Task	Command
Save	C-x C-s
Exit	C-x C-c
Open file	C-x C-f
Cut	C-k
Undo	C-/

Emacs has modes:

- C-mode
- Python-mode
- Org-mode
- Terminal-mode

It also works like an IDE.

---

6. Editors Usage in C Programming

Create a C file

vi hello.c

Write code

Press i and type:

```
#include <stdio.h>


int main() {
   printf("Hello Linux\n");
   return 0;
```

}

Save and exit

:wq

Compile

gcc hello.c -o hello

Run

./hello

---

7. Summary

Editor	Features	Suitable For
vi	Basic, fast	Servers, minimal systems
vim	Improved, powerful	Coding, scripting
emacs	Advanced, customizable	Developers, researchers
nano	Very simple	Beginners

ii) Exposure to Turbo C, gcc

Exposure to Turbo C and GCC

1. What is Turbo C?

Turbo C is an old C compiler and IDE developed by Borland.
It was mainly used in MS-DOS–based systems.

Features of Turbo C

- Blue screen IDE
- Built-in editor and compiler
- Runs in 16-bit environment
- Uses outdated header files (like conio.h)

Limitations

- Not suitable for modern systems
- Does not follow modern C standards
- Limited memory
- 16-bit architecture, not used in the industry now

Typical Turbo C Window

- **Editor** – where you type code
- **Compile/Run** – using menu or shortcut keys

Turbo C Shortcuts

Action	**Shortcut**
Compile	Alt + C + C
Run	Ctrl + F9
View output	Alt + F5
Save	F2

Sample Turbo C Program

```
#include <stdio.h>
#include <conio.h>

void main() {
   clrscr();
```

```
    printf("Hello from Turbo C!");
    getch();
}
```

(Note: conio.h and clrscr() are outdated and not available in modern C.)

---

2. What is GCC?

GCC stands for **GNU Compiler Collection**.
It is the standard C/C++ compiler used in:

- Linux
- Modern C programming
- Industry, academics, systems programming
- Embedded systems

Advantages of GCC

- Follows modern C standards (C11, C17, etc.)
- Runs on Linux, Windows, macOS
- Open-source and widely used
- Produces faster executables
- Better error messages

Basic GCC Compilation Command

```
gcc program.c -o output
```

Run the program

```
./output
```

Sample C Program for GCC

```
#include <stdio.h>

int main() {
    printf("Hello from GCC!\n");
    return 0;
}
```

Key GCC Options

Option	Meaning
-o output	Set output file name
-Wall	Show all warnings
-g	Add debugging info
-O2	Optimization

Example:

gcc -Wall -g -o test test.c

---

3. Turbo C vs GCC (Simple Comparison)

Feature	Turbo C	GCC
Environment	DOS-based	Linux/Windows/macOS
Architecture	16-bit	32/64-bit
Headers	Old (conio.h)	Modern C headers
Industry use	Almost none	Widely used
Speed	Slow	Fast
Standard	Pre-ANSI	ANSI/ISO C standards
IDE	Inbuilt	Terminal + any editor

---

4. Why GCC is used today instead of Turbo C?

- Turbo C is outdated
- No modern system support
- Cannot compile latest C programs
- GCC supports all current standards
- Better for Linux, programming labs, projects, real development

iii) Writing simple programs using printf(), scanf()

Writing Simple Programs Using printf() and scanf()

1. What is printf()?

printf() is used to **display output** on the screen.

Example:

printf("Hello World");

Common Format Specifiers

Data Type	Format
int	%d
float	%f
char	%c
string	%s

---

2. What is scanf()?

scanf() is used to **take input from the user**.

Example:

scanf("%d", &num);

Note:

- Always use **&** (address operator) for variables in scanf() (except strings)

---

3. Simple C Programs

Program 1: Read and print an integer

```
#include <stdio.h>

int main() {
   int num;

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

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

    printf("You entered: %d", num);

    return 0;
}
```

Program 2: Add two numbers

```
#include <stdio.h>

int main() {
    int a, b, sum;

    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

    sum = a + b;

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

    return 0;
}
```

Program 3: Read a character

```
#include <stdio.h>

int main() {
    char ch;

    printf("Enter a character: ");
```

```
    scanf("%c", &ch);

    printf("You entered: %c", ch);

    return 0;
}
```

Program 4: Read and print a float

```
#include <stdio.h>

int main() {
    float marks;

    printf("Enter marks: ");
    scanf("%f", &marks);

    printf("Marks = %.2f", marks);

    return 0;
}
```

Program 5: Read a string

```
#include <stdio.h>

int main() {
    char name[20];

    printf("Enter your name: ");
    scanf("%s", name);   // No & for strings
```

```
    printf("Hello %s", name);

    return 0;
}
```

Program 6: Calculate area of a circle

```
#include <stdio.h>

int main() {
    float r, area;

    printf("Enter radius: ");
    scanf("%f", &r);

    area = 3.14 * r * r;

    printf("Area = %.2f", area);

    return 0;
}
```

Program 7: Simple calculator (addition, subtraction)

```
#include <stdio.h>

int main() {
    int a, b;

    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
```

```
    printf("Addition = %d\n", a + b);
    printf("Subtraction = %d\n", a - b);

    return 0;
}
```

WEEK 2

Tutorial 2: Problem-solving using Algorithms and Flow charts.

Lab 1:

i) Sum and average of 3 numbers

```
#include <stdio.h>

int main() {
    int a, b, c;
    float sum, avg;

    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

    sum = a + b + c;
    avg = sum / 3;

    printf("Sum = %.2f\n", sum);
    printf("Average = %.2f\n", avg);

    return 0;
}
```

Output

```
Enter three numbers: 4 5 7
Sum = 16.00
Average = 5.33
```

ii) Conversion of Fahrenheit to Celsius and vice versa

1. Fahrenheit to Celsius Conversion

Formula

$$C = \frac{5}{9} \times (F - 32)$$

```
#include <stdio.h>

int main() {
    float f, c;

    printf("Enter temperature in Fahrenheit: ");
    scanf("%f", &f);

    c = (5.0 / 9.0) * (f - 32);

    printf("Temperature in Celsius = %.2f\n", c);

    return 0;
}
```

Output

```
Enter temperature in Fahrenheit: 5
Temperature in Celsius = -15.00
```

2. Celsius to Fahrenheit Conversion

Formula

$$F = (C \times 9/5) + 32$$

```
#include <stdio.h>

int main() {
    float c, f;

    printf("Enter temperature in Celsius: ");
    scanf("%f", &c);

    f = (c * 9.0 / 5.0) + 32;

    printf("Temperature in Fahrenheit = %.2f\n", f);

    return 0;
}
```

Output

```
Enter temperature in Celsius: 108
Temperature in Fahrenheit = 226.40
```

iii) Simple interest calculation

Simple Interest Calculation in C

Formula

$$SI = \frac{P \times T \times R}{100}$$

Where:

- **P** = Principal amount
- **T** = Time (in years)
- **R** = Rate of interest

```
#include <stdio.h>

int main() {
    float P, T, R, SI;

    printf("Enter Principal amount: ");
    scanf("%f", &P);

    printf("Enter Time in years: ");
    scanf("%f", &T);

    printf("Enter Rate of interest: ");
    scanf("%f", &R);

    SI = (P * T * R) / 100;

    printf("Simple Interest = %.2f\n", SI);

    return 0;
}
```

Output

```
Enter Principal amount: 41
Enter Time in years: 1
Enter Rate of interest: 1
Simple Interest = 0.41
```

WEEK 3

Tutorial 3: Variable types and type conversions:

Lab 3: Simple computational problems using arithmetic expressions.

i) Finding the square root of a given number

Square Root of a Number in C

You can use the built-in function **sqrt()** from the **math.h** library.

```
#include <stdio.h>
#include <math.h>

int main() {
    double num, result;

    printf("Enter a number: ");
    scanf("%lf", &num);

    result = sqrt(num);

    printf("Square root = %.2lf\n", result);

    return 0;
}
```

Output

```
Enter a number: 41
Square root = 6.40
```

ii) Finding compound interest

Compound Interest Calculation in C

Formula

$$A = P(1 + \frac{R}{100})^T$$
$$CI = A - P$$

Where:

- **P** = Principal
- **R** = Rate of interest
- **T** = Time (years)
- **A** = Amount
- **CI** = Compound Interest

```
#include <stdio.h>
#include <math.h>

int main() {
   float P, R, T, A, CI;

   printf("Enter Principal amount: ");
   scanf("%f", &P);

   printf("Enter Rate of interest: ");
   scanf("%f", &R);

   printf("Enter Time in years: ");
   scanf("%f", &T);

   A = P * pow((1 + R / 100), T);
```

```
    CI = A - P;

    printf("Amount = %.2f\n", A);
    printf("Compound Interest = %.2f\n", CI);

    return 0;
}
```

Output

```
Enter Principal amount: 411
Enter Rate of interest: 5
Enter Time in years: 12
Amount = 738.10
Compound Interest = 327.10
```

iii) Area of a triangle using heron's formulae

Area of Triangle Using Heron's Formula

Formula

If the sides are **a**, **b**, and **c**,
First find semi-perimeter:

$$s = \frac{a + b + c}{2}$$

Then area:

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

```
#include <stdio.h>
#include <math.h>

int main() {
   float a, b, c, s, area;

   printf("Enter three sides of the triangle: ");
   scanf("%f %f %f", &a, &b, &c);

   s = (a + b + c) / 2;

   area = sqrt(s * (s - a) * (s - b) * (s - c));

   printf("Area of the triangle = %.2f\n", area);

   return 0;
}
```

Output

```
Enter three sides of the triangle: 4 5 7
Area of the triangle = 9.80
```

iv) Distance travelled by an object

the **distance travelled by an object**, usually using the physics formula:

$$s = ut + \frac{1}{2}at^2$$

Where:

- **u** = initial velocity
- **a** = acceleration
- **t** = time
- **s** = distance travelled

```
#include <stdio.h>

int main() {
    float u, a, t, s;

    printf("Enter initial velocity (u): ");
    scanf("%f", &u);

    printf("Enter acceleration (a): ");
    scanf("%f", &a);

    printf("Enter time (t): ");
    scanf("%f", &t);

    s = (u * t) + (0.5 * a * t * t);

    printf("Distance travelled = %.2f units\n", s);
```

```
    return 0;
}
```

Output

```
Enter initial velocity (u): 41
Enter acceleration (a): 5
Enter time (t): 6
Distance travelled = 336.00 units
```

UNIT II

WEEK 4

Tutorial4:

Lab4:

i) Evaluate the following expressions.

a. A+B*C+(D*E) + F*G

Expression:

A + B * C + (D * E) + F * G

Since no values are given, we will **evaluate step-by-step using operator precedence**.

✅ **Operator Precedence in C**

1. **Parentheses ( )**
2. **Multiplication *, Division /**
3. **Addition +, Subtraction -**

```
#include <stdio.h>

int main() {
   float A, B, C, D, E, F, G, result;

   printf("Enter values of A, B, C, D, E, F, G: ");
   scanf("%f %f %f %f %f %f %f", &A, &B, &C, &D, &E, &F, &G);

   result = A + (B * C) + (D * E) + (F * G);

   printf("Result of A + B*C + (D*E) + F*G = %.2f\n", result);

   return 0;
}
```

Output

```
Enter values of A, B, C, D, E, F, G: 2 47 8  9  11  2 4  12
Result of A + B*C + (D*E) + F*G = 485.00
```

b) A/B*C-B+A*D/3

```c
#include <stdio.h>

int main() {
    float A, B, C, D, result;

    printf("Enter values of A, B, C, D: ");
    scanf("%f %f %f %f", &A, &B, &C, &D);

    result = (A / B) * C - B + (A * D) / 3;

    printf("Result of A/B*C - B + A*D/3 = %.2f\n", result);

    return 0;
}
```

Output

```
Enter values of A, B, C, D: 4 5 2 1
Result of A/B*C - B + A*D/3 = -2.07
```

c) A+++B---A

```
#include <stdio.h>

int main() {
    int A, B, result;

    printf("Enter values of A and B: ");
    scanf("%d %d", &A, &B);

    result = A++ + B-- - A;

    printf("Result = %d\n", result);
    printf("Final value of A = %d\n", A);
    printf("Final value of B = %d\n", B);

    return 0;
}
```

Output

```
Enter values of A and B: 4 77
Result = 76
Final value of A = 5
Final value of B = 76
```

d) J= (i++) + (++i)

```
#include <stdio.h>

int main() {
    int i, J;

    printf("Enter value of i: ");
    scanf("%d", &i);

    J = (i++) + (++i);

    printf("Value of J = %d\n", J);
    printf("Final value of i = %d\n", i);

    return 0;
}
```

Output

```
Enter value of i: 5
Value of J = 12
Final value of i = 7
```

ii) Find the maximum of three numbers using conditional operator

```
#include <stdio.h>

int main() {
    int a, b, c, max;

    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

    max = (a > b) ? ((a > c) ? a : c)
                  : ((b > c) ? b : c);

    printf("Maximum = %d\n", max);

    return 0;
}
```

Output

```
Enter three numbers: 1 4 5
Maximum = 5
```

iii) Take marks of 5 subjects in integers, and find the total, average in float

```
#include <stdio.h>

int main() {
    int m1, m2, m3, m4, m5;
    int total;
    float average;

    printf("Enter marks of 5 subjects: ");
    scanf("%d %d %d %d %d", &m1, &m2, &m3, &m4, &m5);

    total = m1 + m2 + m3 + m4 + m5;
    average = total / 5.0;   // 5.0 for float division

    printf("Total = %d\n", total);
    printf("Average = %.2f\n", average);

    return 0;
}
```

Output

```
Enter marks of 5 subjects: 55 44 77 11 2
Total = 189
Average = 37.80
```

WEEK 5

Tutorial 5:

Lab 5:

i) Write a C program to find the max and min of four numbers using if-else.

```
#include <stdio.h>

int main() {
    int a, b, c, d;
    int max, min;

    printf("Enter four numbers: ");
    scanf("%d %d %d %d", &a, &b, &c, &d);

    // Finding Maximum
    max = a;
    if (b > max) max = b;
    if (c > max) max = c;
    if (d > max) max = d;

    // Finding Minimum
    min = a;
    if (b < min) min = b;
    if (c < min) min = c;
    if (d < min) min = d;

    printf("Maximum = %d\n", max);
    printf("Minimum = %d\n", min);

    return 0;
```

}

Output

```
Enter four numbers: 4 1 2 8
Maximum = 8
Minimum = 1
```

ii) Write a C program to generate electricity bill.

This program calculates the bill based on **units consumed**.

You can change the rates as per your syllabus.
I am using a common slab system:

- **0 – 100 units** → ₹1.50 per unit
- **101 – 200 units** → ₹2.50 per unit
- **201 – 300 units** → ₹3.50 per unit
- **Above 300 units** → ₹5.00 per unit

```
#include <stdio.h>

int main() {
    int units;
    float bill;

    printf("Enter units consumed: ");
    scanf("%d", &units);

    if (units <= 100) {
        bill = units * 1.50;
    }
    else if (units <= 200) {
        bill = (100 * 1.50) + (units - 100) * 2.50;
    }
    else if (units <= 300) {
        bill = (100 * 1.50) + (100 * 2.50) + (units - 200) * 3.50;
    }
    else {
        bill = (100 * 1.50) + (100 * 2.50) + (100 * 3.50) + (units - 300) * 5.00;
    }
```

```
    printf("Electricity Bill = Rs. %.2f\n", bill);

    return 0;
}
```

Output

```
Enter units consumed: 44
Electricity Bill = Rs. 66.00
```

iii) Find the roots of the quadratic equation.

A quadratic equation is:

$$ax^2 + bx + c = 0$$

The discriminant is:

$$D = b^2 - 4ac$$

- If **D > 0** → two real and different roots
- If **D = 0** → two real and equal roots
- If **D < 0** → two complex roots

```
#include <stdio.h>
#include <math.h>

int main() {
   float a, b, c, D, root1, root2, real, imag;

   printf("Enter values of a, b, c: ");
   scanf("%f %f %f", &a, &b, &c);

   D = (b * b) - (4 * a * c);

   if (D > 0) {
      // Two real and different roots
      root1 = (-b + sqrt(D)) / (2 * a);
      root2 = (-b - sqrt(D)) / (2 * a);
      printf("Roots are Real and Different.\n");
      printf("Root 1 = %.2f\n", root1);
      printf("Root 2 = %.2f\n", root2);
```

```
    }
    else if (D == 0) {
        // Two real and equal roots
        root1 = root2 = -b / (2 * a);
        printf("Roots are Real and Equal.\n");
        printf("Root 1 = Root 2 = %.2f\n", root1);
    }
    else {
        // Complex roots
        real = -b / (2 * a);
        imag = sqrt(-D) / (2 * a);
        printf("Roots are Complex.\n");
        printf("Root 1 = %.2f + %.2fi\n", real, imag);
        printf("Root 2 = %.2f - %.2fi\n", real, imag);
    }

    return 0;
}
```

Output

```
Enter values of a, b, c: 11 22 4
Roots are Real and Different.
Root 1 = -0.20
Root 2 = -1.80
```

iv) Write a C program to simulate a calculator using switch case.

```
#include <stdio.h>

int main() {
    float a, b, result;
    char op;

    printf("Enter first number: ");
    scanf("%f", &a);

    printf("Enter an operator (+, -, *, /): ");
    scanf(" %c", &op);   // space before %c to avoid input issues

    printf("Enter second number: ");
    scanf("%f", &b);

    switch (op) {
        case '+':
            result = a + b;
            printf("Result = %.2f\n", result);
            break;

        case '-':
            result = a - b;
            printf("Result = %.2f\n", result);
            break;

        case '*':
```

```
            result = a * b;
            printf("Result = %.2f\n", result);
            break;

        case '/':
            if (b != 0) {
                result = a / b;
                printf("Result = %.2f\n", result);
            } else {
                printf("Error: Division by zero not allowed.\n");
            }
            break;

        default:
            printf("Invalid operator!\n");
    }

    return 0;
}
```


Output

```
Enter first number: 4
Enter an operator (+, -, *, /): +
Enter second number: 7
Result = 11.00
```

v) Write a C program to find the given year is a leap year or not.

```
#include <stdio.h>

int main() {
    int year;

    printf("Enter a year: ");
    scanf("%d", &year);

    if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0)) {
        printf("%d is a Leap Year.\n", year);
    }
    else {
        printf("%d is NOT a Leap Year.\n", year);
    }

    return 0;
}
```

Output	Output
Enter a year: 2025 2025 is NOT a Leap Year.	Enter a year: 2024 2024 is a Leap Year.

WEEK 6

Tutorial 6

Lab 6

i) Find the factorial of given number using any loop.

```
#include <stdio.h>

int main() {
    int n, i;
    long long fact = 1;   // long long for large values

    printf("Enter a number: ");
    scanf("%d", &n);

    if (n < 0) {
        printf("Factorial of negative numbers is not possible.\n");
    } else {
        for (i = 1; i <= n; i++) {
            fact = fact * i;
        }

        printf("Factorial of %d = %lld\n", n, fact);
    }

    return 0;
}
```


Output

```
Enter a number: 52
Factorial of 52 = -8452693550620999680
```

ii) Find the given number is a prime or not.

```
#include <stdio.h>

int main() {
   int n, i, flag = 0;

   printf("Enter a number: ");
   scanf("%d", &n);

   if (n <= 1) {
      printf("%d is NOT a prime number.\n", n);
   } else {
      for (i = 2; i <= n / 2; i++) {
         if (n % i == 0) {
            flag = 1;
            break;
         }
      }

      if (flag == 0)
         printf("%d is a Prime number.\n", n);
      else
         printf("%d is NOT a Prime number.\n", n);
   }

   return 0;
}
```

Output

```
Enter a number: 2025
2025 is NOT a Prime number.
```

Output

```
Enter a number: 3
3 is a Prime number.
```

ii) Compute sine and cos series.

✅ **Formulas**

Sine Series (sin x)

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Cosine Series (cos x)

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

```
#include <stdio.h>
#include <math.h>

int main() {
    int n, i;
    float x, term, sine = 0, cosine = 0;

    printf("Enter the value of x in degrees: ");
    scanf("%f", &x);

    printf("Enter number of terms: ");
    scanf("%d", &n);

    // Convert degrees to radians
    x = x * (3.14159 / 180);

    // Sine series
    term = x;
    for (i = 1; i <= n; i++) {
        sine += term;
        term = -term * x * x / ((2 * i) * (2 * i + 1));
    }
```

```
    // Cosine series
    term = 1;
    for (i = 1; i <= n; i++) {
        cosine += term;
        term = -term * x * x / ((2 * i - 1) * (2 * i));
    }

    printf("Sine(%f) = %f\n", x, sine);
    printf("Cosine(%f) = %f\n", x, cosine);

    return 0;
}
```

Output

```
Enter the value of x in degrees: 2025
Enter number of terms: 2024
Sine(35.342888) = -7648640.500000
Cosine(35.342888) = -1224712.500000
```

iii) Checking a number palindrome

```
#include <stdio.h>

int main() {
    int num, original, reversed = 0, digit;

    printf("Enter a number: ");
    scanf("%d", &num);

    original = num;  // store original number

    while (num > 0) {
        digit = num % 10;           // get last digit
        reversed = reversed * 10 + digit; // build reversed number
        num = num / 10;             // remove last digit
    }

    if (original == reversed)
        printf("%d is a Palindrome.\n", original);
    else
        printf("%d is NOT a Palindrome.\n", original);

    return 0;
}
```


Output

```
Enter a number: 2025
2025 is NOT a Palindrome.
```

Output

```
Enter a number: 202
202 is a Palindrome.
```

v) Construct a pyramid of numbers.

```
#include <stdio.h>

int main() {
    int n, i, j;

    printf("Enter number of rows: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++) {      // outer loop for rows
        for (j = 1; j <= i; j++) {  // inner loop for printing numbers
            printf("%d ", j);
        }
        printf("\n");
    }

    return 0;
}
```

Output

```
Enter number of rows: 4
1
1 2
1 2 3
1 2 3 4
```

UNIT III

WEEK 7:

Tutorial 7: 1 D Arrays: searching.

Lab 7:1D Array manipulation, linear search

i) Find the min and max of a 1-D integer array.

```
#include <stdio.h>

int main() {
    int n, i;
    int arr[100];
    int min, max;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Initialize min and max
    min = max = arr[0];

    // Find min and max
    for (i = 1; i < n; i++) {
        if (arr[i] > max)
            max = arr[i];

        if (arr[i] < min)
```

```
            min = arr[i];
        }

        printf("Maximum = %d\n", max);
        printf("Minimum = %d\n", min);

        return 0;
}
```

Output

```
Enter number of elements: 9
Enter 9 integers:
11
22
77
5
0
5
124
2025
5567
Maximum = 5567
Minimum = 0
```

ii) Perform linear search on1D array.

```
#include <stdio.h>

int main() {
    int arr[100], n, i, key, found = 0;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the element to search: ");
    scanf("%d", &key);

    // Linear Search
    for (i = 0; i < n; i++) {
        if (arr[i] == key) {
            found = 1;
            break;
        }
    }

    if (found == 1)
        printf("Element %d found at position %d.\n", key, i + 1);
    else
        printf("Element %d not found in the array.\n", key);
```

```
    return 0;
}
```

Output

```
Enter number of elements: 5
Enter 5 elements:
11
44
2025
71
0
Enter the element to search: 5
Element 5 not found in the array.
```

iii) The reverse of a 1D integer array

```
#include <stdio.h>

int main() {
    int arr[100], n, i;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Array in Reverse Order:\n");
    for (i = n - 1; i >= 0; i--) {
        printf("%d ", arr[i]);
    }

    printf("\n");
    return 0;
}
```

Output

```
Enter number of elements: 5
Enter 5 elements:
4
7
12
2025
001
Array in Reverse Order:
1 2025 12 7 4
```

iv) Find 2's complement of the given binary number.

```
#include <stdio.h>
#include <string.h>

int main() {
    char bin[100];
    int i, len;

    printf("Enter a binary number: ");
    scanf("%s", bin);

    len = strlen(bin);

    // Step 1: Find 1's Complement
    for (i = 0; i < len; i++) {
        if (bin[i] == '0')
            bin[i] = '1';
        else if (bin[i] == '1')
            bin[i] = '0';
    }

    // Step 2: Add 1 to get 2's Complement
    for (i = len - 1; i >= 0; i--) {
        if (bin[i] == '0') {
            bin[i] = '1';
            break;
        } else {
```

```
            bin[i] = '0';
        }
    }

    printf("2's Complement = %s\n", bin);

    return 0;
}
```

Output

```
Enter a binary number: 1001
2's Complement = 0111
```

v) Eliminate duplicate elements in an array

```
#include <stdio.h>

int main() {
    int arr[100], n, i, j, k;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Removing duplicates
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                // Shift elements left to remove duplicate
                for (k = j; k < n - 1; k++) {
                    arr[k] = arr[k + 1];
                }
                n--;       // decrease size
                j--;       // check same index again
            }
        }
    }
```

```
    printf("Array after eliminating duplicates:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    printf("\n");
    return 0;
}
```

Output

```
Enter number of elements: 5
Enter 5 elements:
2025
014
025
0
2025
Array after eliminating duplicates:
2025 14 25 0
```

WEEK 8:

Tutorial 8: 2 D arrays, sorting and Strings

Lab 8: Matrix problems, String operations, Bubble sort

i) Addition of two matrices

```
#include <stdio.h>

int main() {
    int r, c, i, j;
    int A[10][10], B[10][10], Sum[10][10];

    printf("Enter number of rows: ");
    scanf("%d", &r);

    printf("Enter number of columns: ");
    scanf("%d", &c);

    // Input for Matrix A
    printf("Enter elements of Matrix A:\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            scanf("%d", &A[i][j]);
        }
    }

    // Input for Matrix B
    printf("Enter elements of Matrix B:\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            scanf("%d", &B[i][j]);
        }
```

```
    }

    // Addition of Matrices
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            Sum[i][j] = A[i][j] + B[i][j];
        }
    }

    // Output the Result
    printf("Sum of the Matrices:\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            printf("%d ", Sum[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output

```
Enter number of rows: 3
Enter number of columns: 3
Enter elements of Matrix A:
12 45 78
2 5 89
36 43 74
Enter elements of Matrix B:
15 59 654
75 53 54
48
78 96
Sum of the Matrices:
27 104 732
77 58 143
84 121 170
```

ii) Multiplication two matrices

```
#include <stdio.h>

int main() {
    int r1, c1, r2, c2;
    int A[10][10], B[10][10], M[10][10];
    int i, j, k;

    // Input order for Matrix A
    printf("Enter rows and columns of Matrix A: ");
    scanf("%d %d", &r1, &c1);

    // Input order for Matrix B
    printf("Enter rows and columns of Matrix B: ");
    scanf("%d %d", &r2, &c2);

    // Check for multiplication condition
    if (c1 != r2) {
        printf("Matrix multiplication is NOT possible.\n");
        return 0;
    }

    // Input Matrix A
    printf("Enter elements of Matrix A:\n");
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c1; j++) {
            scanf("%d", &A[i][j]);
```

```
    }
}

// Input Matrix B
printf("Enter elements of Matrix B:\n");
for (i = 0; i < r2; i++) {
    for (j = 0; j < c2; j++) {
        scanf("%d", &B[i][j]);
    }
}

// Initialize result matrix to 0
for (i = 0; i < r1; i++) {
    for (j = 0; j < c2; j++) {
        M[i][j] = 0;
    }
}

// Matrix Multiplication
for (i = 0; i < r1; i++) {
    for (j = 0; j < c2; j++) {
        for (k = 0; k < c1; k++) {
            M[i][j] += A[k][j] * B[k][j];
        }
    }
}

// Output Result
printf("Product of the Matrices:\n");
for (i = 0; i < r1; i++) {
```

```
        for (j = 0; j < c2; j++) {
            printf("%d ", M[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output

```
Enter rows and columns of Matrix A: 3
3
Enter rows and columns of Matrix B: 3 3
Enter elements of Matrix A:
32 54 78
65 48 12
55 22 11
Enter elements of Matrix B:
33 66 77
444 555 999
123 456 789
Product of the Matrices:
36681 40236 26673
36681 40236 26673
36681 40236 26673
```

iii) Sort array elements using bubble sort

```
#include <stdio.h>

int main() {
    int arr[100], n, i, j, temp;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Bubble Sort
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // swap elements
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    printf("Sorted array:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
```

```
    }

    printf("\n");
    return 0;
}
```

Output

```
Enter number of elements: 8
Enter 8 elements:
41 52 6 0 112 4010 01222 85553
Sorted array:
0 6 41 52 112 1222 4010 85553
```

iv) Concatenate two strings without built-in functions

```
#include <stdio.h>

int main() {
    char str1[100], str2[100];
    int i = 0, j = 0;

    printf("Enter first string: ");
    scanf("%s", str1);

    printf("Enter second string: ");
    scanf("%s", str2);

    // Move i to the end of str1
    while (str1[i] != '\0') {
        i++;
    }

    // Copy str2 into str1
    while (str2[j] != '\0') {
        str1[i] = str2[j];
        i++;
        j++;
    }

    // Add null terminator at the end
    str1[i] = '\0';

    printf("Concatenated String = %s\n", str1);
```

```
    return 0;
}
```

Output

```
Enter first string: hello bro
Enter second string: Concatenated String = hellobro
```

v) Reverse a string using built-in and without built-in string functions

```
#include <stdio.h>
#include <string.h>

int main() {
    char str[100];

    printf("Enter a string: ");
    scanf("%s", str);

    strrev(str);   // built-in function (Turbo C)

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

    return 0;
}
```

Output

```
Enter a string: hello bro
Reversed string = olleh
```

UNIT IV

Week 9

Tutorial 9: Pointers, structures and dynamic memory allocation

Lab 9: Pointers and structures, memory dereference.

i) Write a C program to find the sum of a 1D array using malloc()

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *arr, n, i, sum = 0;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    // Allocating memory dynamically
    arr = (int *)malloc(n * sizeof(int));

    if (arr == NULL) {
        printf("Memory not allocated!\n");
        return 0;
    }

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        sum += arr[i];
    }
```

```
    printf("Sum of array elements = %d\n", sum);

    free(arr);  // release memory

    return 0;
}
```

Output

```
Enter number of elements: 4
Enter 4 elements:
44 55 78 3
Sum of array elements = 180
```

ii) Write a C program to find the total, average of n students using structures

```
#include <stdio.h>

struct Student {
    int marks;
};

int main() {
    struct Student s[100];
    int n, i, total = 0;
    float average;

    printf("Enter number of students: ");
    scanf("%d", &n);

    printf("Enter marks of %d students:\n", n);
    for (i = 0; i < n; i++) {
        printf("Marks of student %d: ", i + 1);
        scanf("%d", &s[i].marks);
        total += s[i].marks;
    }

    average = total / (float)n;

    printf("\nTotal Marks = %d\n", total);
    printf("Average Marks = %.2f\n", average);

    return 0;
}
```

Output

```
Enter number of students: 5
Enter marks of 5 students:
Marks of student 1: 54
Marks of student 2: 55
Marks of student 3: 2
Marks of student 4: 0
Marks of student 5: 100

Total Marks = 211
Average Marks = 42.20
```

iii) Enter n students data using calloc() and display failed students list

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *marks;
    int n, i;

    printf("Enter number of students: ");
    scanf("%d", &n);

    // Allocate memory using calloc()
    marks = (int *)calloc(n, sizeof(int));

    if (marks == NULL) {
        printf("Memory not allocated!\n");
        return 0;
    }

    // Accept marks
    printf("Enter marks of %d students:\n", n);
    for (i = 0; i < n; i++) {
        printf("Marks of student %d: ", i + 1);
        scanf("%d", &marks[i]);
    }

    // Display failed students
    printf("\nFailed Students (Marks < 35):\n");
    int found = 0;
```

```
    for (i = 0; i < n; i++) {
        if (marks[i] < 35) {
            printf("Student %d - Marks: %d\n", i + 1, marks[i]);
            found = 1;
        }
    }

    if (!found)
        printf("No student has failed.\n");

    free(marks);  // release memory

    return 0;
}
```

Output

```
Enter number of students: 5
Enter marks of 5 students:
Marks of student 1: 11
Marks of student 2: 45
Marks of student 3: 70
Marks of student 4: 21
Marks of student 5: 1

Failed Students (Marks < 35):
Student 1 - Marks: 11
Student 4 - Marks: 21
Student 5 - Marks: 1
```

iv) Read student name and marks from the command line and display the student details along with the total.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int i, total = 0;

    if (argc < 3) {
        printf("Usage: %s <name> <mark1> <mark2> ...\n", argv[0]);
        return 0;
    }

    // First argument is name
    printf("Student Name: %s\n", argv[1]);

    // Remaining arguments are marks
    for (i = 2; i < argc; i++) {
        total += atoi(argv[i]);  // convert string to integer
    }

    printf("Total Marks = %d\n", total);

    return 0;
}
```

```
Student Name: Ramesh
Total Marks = 380
```

v) Write a C program to implement realloc()

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *arr;
    int n, newSize, i;

    // Step 1: Allocate initial memory
    printf("Enter initial number of elements: ");
    scanf("%d", &n);

    arr = (int *)malloc(n * sizeof(int));

    if (arr == NULL) {
        printf("Memory not allocated!\n");
        return 0;
    }

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Step 2: Reallocate memory using realloc()
    printf("\nEnter new size of array: ");
    scanf("%d", &newSize);

    arr = (int *)realloc(arr, newSize * sizeof(int));
```

```
    if (arr == NULL) {
        printf("Memory reallocation failed!\n");
        return 0;
    }

    // Step 3: Read extra elements if size increased
    if (newSize > n) {
        printf("Enter %d more elements:\n", newSize - n);
        for (i = n; i < newSize; i++) {
            scanf("%d", &arr[i]);
        }
    }

    // Step 4: Display final array
    printf("\nFinal Array Elements:\n");
    for (i = 0; i < newSize; i++) {
        printf("%d ", arr[i]);
    }

    printf("\n");

    free(arr);  // Free memory
    return 0;
}
```

Output

```
Enter initial number of elements: 5
Enter 5 elements:
11 45 1 5 0

Enter new size of array: 4

Final Array Elements:
11 45 1 5
```

WEEK 10

LAB 10

i) Create and display a singly linked list using self-referential structure.

```
#include <stdio.h>
#include <stdlib.h>

/* Self-referential structure for node */
struct Node {
    int data;
    struct Node *next;
};

/* Function to create a new node with given value */
struct Node* createNode(int value) {
    struct Node *newNode = (struct Node*) malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}

/* Function to append a node at the end of list */
void appendNode(struct Node **headRef, int value) {
    struct Node *newNode = createNode(value);

    if (*headRef == NULL) {
```

```
        *headRef = newNode;     // list was empty
        return;
    }

    struct Node *temp = *headRef;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = newNode;
}

/* Function to display the linked list */
void displayList(struct Node *head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

    printf("Linked List: ");
    while (head != NULL) {
        printf("%d", head->data);
        if (head->next != NULL) printf(" -> ");
        head = head->next;
    }
    printf("\n");
}

/* Free all nodes */
void freeList(struct Node *head) {
    struct Node *tmp;
```

```
    while (head != NULL) {
        tmp = head;
        head = head->next;
        free(tmp);
    }
}

int main() {
    struct Node *head = NULL;
    int n, i, val;

    printf("Enter number of nodes: ");
    if (scanf("%d", &n) != 1 || n < 0) {
        printf("Invalid input.\n");
        return 1;
    }

    for (i = 0; i < n; i++) {
        printf("Enter value for node %d: ", i + 1);
        scanf("%d", &val);
        appendNode(&head, val);
    }

    displayList(head);

    freeList(head);
    return 0;
}
```

Output

```
Enter number of nodes: 4
Enter value for node 1: 44
Enter value for node 2: 2
Enter value for node 3: 8
Enter value for node 4: 145
Linked List: 44 -> 2 -> 8 -> 145
```

ii) Demonstrate the differences between structures and unions using a C program.

```
#include <stdio.h>

/* Structure with 3 members */
struct StudentStruct {
    int roll;
    float marks;
    char grade;
};

/* Union with same 3 members */
union StudentUnion {
    int roll;
    float marks;
    char grade;
};

int main() {
    struct StudentStruct s;
    union StudentUnion u;

    printf("Size of Structure = %lu bytes\n", sizeof(s));
    printf("Size of Union     = %lu bytes\n\n", sizeof(u));

    // Assign values to structure
    s.roll = 101;
    s.marks = 85.5;
    s.grade = 'A';
```

```
    // Assign values to union
    u.roll = 101;
    u.marks = 85.5;
    u.grade = 'A';

    printf("Structure Values:\n");
    printf("Roll  = %d\n", s.roll);
    printf("Marks = %.2f\n", s.marks);
    printf("Grade = %c\n\n", s.grade);

    printf("Union Values (Only last assigned is valid):\n");
    printf("Roll  = %d\n", u.roll);     // invalid after assigning grade
    printf("Marks = %.2f\n", u.marks);  // invalid after assigning grade
    printf("Grade = %c\n", u.grade);    // only this is valid

    return 0;
}
```


Output

```
Size of Structure = 12 bytes
Size of Union     = 4 bytes

Structure Values:
Roll  = 101
Marks = 85.50
Grade = A

Union Values (Only last assigned is valid):
Roll  = 1118502977
Marks = 85.50
Grade = A
```

iii) Write a C program to shift/rotate using bitfields.

```
#include <stdio.h>
#include <stdint.h>

/* union to view an 8-bit byte as individual bitfields and as a whole byte */
typedef union {
    uint8_t byte;
    struct {
        unsigned b0:1;
        unsigned b1:1;
        unsigned b2:1;
        unsigned b3:1;
        unsigned b4:1;
        unsigned b5:1;
        unsigned b6:1;
        unsigned b7:1;
    } bits;
} Byte;

/* print bits using bitfield members (b7..b0) */
void print_bits_bitfield(Byte x) {
    printf("%u%u%u%u %u%u%u%u", x.bits.b7, x.bits.b6, x.bits.b5, x.bits.b4,
                               x.bits.b3, x.bits.b2, x.bits.b1, x.bits.b0);
}

/* print bits using masking (safe and portable) */
void print_bits_mask(uint8_t v) {
    int i;
    for (i = 7; i >= 0; i--) {
```

```
        printf("%u", (v >> i) & 1);
        if (i == 4) printf(" ");
    }
}

/* rotate left by n (0 < n < 8) */
uint8_t rol(uint8_t v, unsigned n) {
    n &= 7; // keep n in 0..7
    return (uint8_t)((v << n) | (v >> (8 - n)));
}

/* rotate right by n (0 < n < 8) */
uint8_t ror(uint8_t v, unsigned n) {
    n &= 7;
    return (uint8_t)((v >> n) | (v << (8 - n)));
}

int main() {
    Byte x;
    unsigned n;
    int choice;

    printf("Enter an 8-bit value (0-255): ");
    if (scanf("%u", &n) != 1 || n > 255) {
        printf("Invalid input.\n");
        return 1;
    }
    x.byte = (uint8_t)n;

    printf("Original byte = %3u\n", x.byte);
```

```
    printf("Bits (bitfield) : "); print_bits_bitfield(x);   printf("\n");
    printf("Bits (mask)     : "); print_bits_mask(x.byte);  printf("\n\n");

    printf("Rotate left by how many positions (1-7)? ");
    if (scanf("%u", &n) != 1) return 1;
    x.byte = rol(x.byte, n);
    printf("\nAfter ROL %u -> byte = %3u\n", n, x.byte);
    printf("Bits (bitfield) : "); print_bits_bitfield(x);   printf("\n");
    printf("Bits (mask)     : "); print_bits_mask(x.byte);  printf("\n\n");

    printf("Rotate right by how many positions (1-7)? ");
    if (scanf("%u", &n) != 1) return 1;
    x.byte = ror(x.byte, n);
    printf("\nAfter ROR %u -> byte = %3u\n", n, x.byte);
    printf("Bits (bitfield) : "); print_bits_bitfield(x);   printf("\n");
    printf("Bits (mask)     : "); print_bits_mask(x.byte);  printf("\n");

    return 0;
}
```

Output

```
Enter an 8-bit value (0-255): 190
Original byte = 190
Bits (bitfield) : 1011 1110
Bits (mask)     : 1011 1110

Rotate left by how many positions (1-7)? 5

After ROL 5 -> byte = 215
Bits (bitfield) : 1101 0111
Bits (mask)     : 1101 0111

Rotate right by how many positions (1-7)? 2

After ROR 2 -> byte = 245
Bits (bitfield) : 1111 0101
Bits (mask)     : 1111 0101
```

iv) Write a C program to copy one structure variable to another structure of the same type.

```
#include <stdio.h>

/* Structure definition */
struct Student {
    int roll;
    float marks;
};

int main() {
    struct Student s1, s2;

    printf("Enter roll number for student 1: ");
    scanf("%d", &s1.roll);

    printf("Enter marks for student 1: ");
    scanf("%f", &s1.marks);

    /* Copying structure s1 into s2 */
    s2 = s1;

    printf("\n--- Student 2 Details (After Copy) ---\n");
    printf("Roll Number: %d\n", s2.roll);
    printf("Marks: %.2f\n", s2.marks);

    return 0;
}
```

Output

```
Enter roll number for student 1: 2496450224
Enter marks for student 1: 45 7 45 51 2

--- Student 2 Details (After Copy) ---
Roll Number: -1798517072
Marks: 45.00
```

Unit V

WEEK 11

Tutorial 11

Lab 11

i) Write a C function to calculate NCR value.

```
#include <stdio.h>

/* Function to calculate factorial */
long long factorial(int n) {
    long long fact = 1;
    int i;
    for (i = 1; i <= n; i++) {
        fact *= i;
    }
    return fact;
}

/* Function to calculate nCr */
long long nCr(int n, int r) {
    return factorial(n) / (factorial(r) * factorial(n - r));
}

int main() {
    int n, r;

    printf("Enter n and r values: ");
    scanf("%d %d", &n, &r);

    if (r > n || n < 0 || r < 0) {
        printf("Invalid input! r should be <= n and both positive.\n");
```

```
    } else {
        printf("nCr = %lld\n", nCr(n, r));
    }

    return 0;
}
```

Output

```
Enter n and r values: 6 3
nCr = 20
```

ii) Write a C function to find the length of a string.

```
#include <stdio.h>

/* Function to find length of a string */
int stringLength(char str[]) {
    int i = 0;
    while (str[i] != '\0') {
        i++;
    }
    return i;
}

int main() {
    char str[100];

    printf("Enter a string: ");
    scanf("%s", str);   // reads a single word

    printf("Length of the string = %d\n", stringLength(str));

    return 0;
}
```

Output

```
Enter a string: hello brother
Length of the string = 5
```

iii) Write a C function to transpose of a matrix.

```
#include <stdio.h>

/* Function to find transpose of a matrix */
void transpose(int a[10][10], int r, int c, int t[10][10]) {
    int i, j;
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            t[j][i] = a[i][j];   // swapping rows and columns
        }
    }
}

int main() {
    int a[10][10], t[10][10];
    int r, c, i, j;

    printf("Enter rows and columns: ");
    scanf("%d %d", &r, &c);

    printf("Enter elements of the matrix:\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            scanf("%d", &a[i][j]);
        }
    }
```

```
    /* Function call */
    transpose(a, r, c, t);

    printf("\nTranspose of the matrix:\n");
    for (i = 0; i < c; i++) {
        for (j = 0; j < r; j++) {
            printf("%d ", t[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output

```
Enter rows and columns: 2
3
Enter elements of the matrix:
1 2 4 5
1 4 5 3

Transpose of the matrix:
1 5
2 1
4 4
```

iv) Write a C function to demonstrate numerical integration of differential equations using Euler's method

```
#include <stdio.h>
#include <math.h>

/* Define the differential equation dy/dt = f(t,y)
   Change this function to solve a different ODE.
   Example used below in comments: f(t,y) = y  (=> exact solution y = e^t)
*/
double f(double t, double y) {
    return y;   // change to any expression, e.g., -2*y + t
}

int main() {
    double t0, y0, t, y, h, t_end;
    int steps, i;

    printf("Euler's Method for dy/dt = f(t,y)\n\n");

    printf("Enter initial t (t0): ");
    if (scanf("%lf", &t0) != 1) return 0;

    printf("Enter initial y (y0): ");
    if (scanf("%lf", &y0) != 1) return 0;

    printf("Enter step size h: ");
    if (scanf("%lf", &h) != 1) return 0;

    printf("Enter final t (t_end): ");
    if (scanf("%lf", &t_end) != 1) return 0;
```

```
if (h <= 0 || t_end <= t0) {
    printf("Invalid inputs: h must be > 0 and t_end > t0\n");
    return 0;
}

// compute number of steps (rounded down)
steps = (int)((t_end - t0) / h);

t = t0;
y = y0;

printf("\nStep\t   t\t\t y_Euler");
// optional: print exact if known (example exact y = exp(t) when f=y)
printf("\t   y_exact\t  error\n");

for (i = 0; i <= steps; i++) {
    double y_exact = exp(t);          // change or remove if exact is not known
    double err = fabs(y - y_exact);

    printf("%3d\t%8.4f\t%12.6f\t%12.6f\t%8.6f\n", i, t, y, y_exact, err);

    // Euler step
    y = y + h * f(t, y);
    t = t + h;
}

// if t_end not exactly reached due to rounding, do one last small step
if (t < t_end + 1e-12) {
    double last_h = t_end - (t - h); // last step to reach t_end exactly
```

```
        if (last_h > 1e-15) {
            // roll back last step: compute y at (t-h) then step with last_h
            // (simple approach: redo from initial — but here for brevity we accept small mismatch)
        }
    }

    printf("\nNote: Change function f(t,y) in the code to solve other ODEs.\n");
    printf("Smaller h gives better accuracy but needs more steps.\n");

    return 0;
}
```

Output

```
Euler's Method for dy/dt = f(t,y)

Enter initial t (t0): 1
Enter initial y (y0): 2
Enter step size h: 3
Enter final t (t_end): 3.2

Step       t         y_Euler       y_exact    error
  0   1.0000        2.000000        2.718282    0.718282

Note: Change function f(t,y) in the code to solve other ODEs.
Smaller h gives better accuracy but needs more steps.
```

Week 12

Lab 12

i) Write a recursive function to generate Fibonacci series.

```
#include <stdio.h>

/* Recursive Fibonacci function */
int fib(int n) {
   if (n == 0)
      return 0;
   if (n == 1)
      return 1;
   return fib(n - 1) + fib(n - 2);
}

int main() {
   int n, i;

   printf("Enter how many Fibonacci terms: ");
   scanf("%d", &n);

   printf("Fibonacci Series: ");
   for (i = 0; i < n; i++) {
      printf("%d ", fib(i));
   }

   printf("\n");
   return 0;
}
```

Output

```
Enter how many Fibonacci terms: 5
Fibonacci Series: 0 1 1 2 3
```

ii) Write a recursive function to find the lcm of two numbers

```
#include <stdio.h>

/* Recursive GCD function (Euclid's method) */
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

/* LCM using recursive GCD */
int lcm(int a, int b) {
    return (a * b) / gcd(a, b);
}

int main() {
    int a, b;

    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

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

    return 0;
}
```

```
Output
Enter two numbers: 45 4
LCM = 180
```

iii) Write a recursive function to find the factorial of a number.

```
#include <stdio.h>

/* Recursive function to find factorial */
long long factorial(int n) {
    if (n == 0 || n == 1)
        return 1;              // base case
    return n * factorial(n - 1); // recursive step
}

int main() {
    int n;

    printf("Enter a number: ");
    scanf("%d", &n);

    if (n < 0)
        printf("Factorial of negative number is not possible.\n");
    else
        printf("Factorial of %d = %lld\n", n, factorial(n));

    return 0;
}
```

Output

```
Enter a number: 55
Factorial of 55 = 6711489344688881664
```

iv) Write a C Program to implement Ackermann function using recursion.

```
#include <stdio.h>

/* Recursive Ackermann function */
int ackermann(int m, int n) {
    if (m == 0)
        return n + 1;
    else if (n == 0)
        return ackermann(m - 1, 1);
    else
        return ackermann(m - 1, ackermann(m, n - 1));
}

int main() {
    int m, n;

    printf("Enter values of m and n: ");
    scanf("%d %d", &m, &n);

    if (m < 0 || n < 0) {
        printf("Ackermann function is only defined for non-negative integers.\n");
        return 0;
    }

    printf("Ackermann(%d, %d) = %d\n", m, n, ackermann(m, n));

    return 0;
}
```

Output

```
Enter values of m and n: 2
3
Ackermann(2, 3) = 9
```

v) Write a recursive function to find the sum of series.

```
#include <stdio.h>

/* Recursive function to find sum of series: 1 + 2 + ... + n */
int sumSeries(int n) {
    if (n == 1)
        return 1;              // base case
    return n + sumSeries(n - 1); // recursive step
}

int main() {
    int n;

    printf("Enter n value: ");
    scanf("%d", &n);

    if (n < 1) {
        printf("Invalid input! n must be >= 1.\n");
    } else {
        printf("Sum of series = %d\n", sumSeries(n));
    }

    return 0;
}
```

Output

```
Enter n value: 5
Sum of series = 15
```

WEEK 13

LAB 13

i) Write a C program to swap two numbers using call by reference.

```
#include <stdio.h>

/* Function to swap using call by reference */
void swap(int *x, int *y) {
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

int main() {
    int a, b;

    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

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

    /* Call by reference */
    swap(&a, &b);

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

    return 0;
}
```

Output

```
Enter two numbers: 5 4

Before Swap: a = 5, b = 4
After Swap : a = 4, b = 5
```

ii) Demonstrate Dangling pointer problem using a C program

```
#include <stdio.h>
#include <stdlib.h>

int* createMemory() {
    int *p = (int*)malloc(sizeof(int));  // allocate memory
    *p = 100;
    return p;   // return pointer
}

int main() {
    int *ptr = createMemory();

    printf("Value before free: %d\n", *ptr);

    free(ptr);  // memory freed → ptr becomes dangling

    // Dangling pointer used here
    printf("Value after free (Dangling Pointer): %d\n", *ptr);

    // Fix: set ptr to NULL
    ptr = NULL;

    return 0;
}
```

Output

```
Value before free: 100
Value after free (Dangling Pointer): 114810
```

iii) Write a C program to copy one string into another using pointer.

```
#include <stdio.h>

int main() {
    char str1[100], str2[100];
    char *p1, *p2;

    printf("Enter a string: ");
    scanf("%s", str1);

    p1 = str1;  // source pointer
    p2 = str2;  // destination pointer

    // Copy characters using pointers
    while (*p1 != '\0') {
        *p2 = *p1;
        p1++;
        p2++;
    }

    // Add null terminator
    *p2 = '\0';

    printf("Copied String = %s\n", str2);

    return 0;
}
```

Output

```
Enter a string: hello brother how are you?
Copied String = hello
```

iv) Write a C program to find no of lowercase, uppercase, digits and other characters using pointers.

```
#include <stdio.h>

int main() {
    char str[200];
    char *p;
    int lower = 0, upper = 0, digits = 0, others = 0;

    printf("Enter a string: ");
    scanf("%[^\n]", str);  // reads full line including spaces

    p = str;  // pointer to the string

    while (*p != '\0') {
        if (*p >= 'a' && *p <= 'z')
            lower++;
        else if (*p >= 'A' && *p <= 'Z')
            upper++;
        else if (*p >= '0' && *p <= '9')
            digits++;
        else
            others++;

        p++; // move pointer to next character
    }

    printf("\nNumber of lowercase characters = %d\n", lower);
    printf("Number of uppercase characters = %d\n", upper);
    printf("Number of digits = %d\n", digits);
```

```
    printf("Number of other characters = %d\n", others);

    return 0;
}
```

Output

```
Enter a string: hello Bro wHY ar eyou?

Number of lowercase characters = 14
Number of uppercase characters = 3
Number of digits = 0
Number of other characters = 5
```

WEEK 14

LAB 14

i) Write a C program to write and read text into a file.

```
#include <stdio.h>

int main() {
    FILE *fp;
    char text[200];

    // Writing to file
    fp = fopen("sample.txt", "w");
    if (fp == NULL) {
        printf("Error opening file for writing!\n");
        return 0;
    }

    printf("Enter text to write into file: ");
    scanf("%[^\n]", text);   // read full line including spaces

    fprintf(fp, "%s", text);

    fclose(fp);  // close after writing

    // Reading from file
    fp = fopen("sample.txt", "r");
    if (fp == NULL) {
        printf("Error opening file for reading!\n");
        return 0;
    }
```

```
    printf("\nReading from file:\n");

    while (fgets(text, sizeof(text), fp) != NULL) {
        printf("%s", text);
    }

    fclose(fp);

    return 0;
}
```

```
Enter text to write into file: Welcome to C Programming

Reading from file:
Welcome to C Programming
```

ii) Write a C program to write and read text into a binary file using fread() and fwrite()

```
#include <stdio.h>
#include <string.h>

int main() {
    FILE *fp;
    char text[200], buffer[200];

    // ---- Writing to binary file ----
    fp = fopen("data.bin", "wb");  // open for binary write
    if (fp == NULL) {
        printf("Error opening file for writing!\n");
        return 0;
    }

    printf("Enter text to write into binary file: ");
    scanf("%[^\n]", text);   // read full line

    // Write text into binary file
    fwrite(text, sizeof(char), strlen(text) + 1, fp);

    fclose(fp);


    // ---- Reading from binary file ----
    fp = fopen("data.bin", "rb");  // open for binary read
    if (fp == NULL) {
        printf("Error opening file for reading!\n");
        return 0;
    }
```

```
    // Read text from file
    fread(buffer, sizeof(char), sizeof(buffer), fp);

    printf("\nReading from binary file:\n%s\n", buffer);

    fclose(fp);

    return 0;
}
```

```
Enter text to write into binary file: Hello Binary File

Reading from binary file:
Hello Binary File
```

iii) Copy the contents of one file to another file.

```
#include <stdio.h>

int main() {
    FILE *fp1, *fp2;
    char ch;

    // Open source file in read mode
    fp1 = fopen("source.txt", "r");
    if (fp1 == NULL) {
        printf("Error opening source file!\n");
        return 0;
    }

    // Open destination file in write mode
    fp2 = fopen("destination.txt", "w");
    if (fp2 == NULL) {
        printf("Error opening destination file!\n");
        fclose(fp1);
        return 0;
    }

    // Copy character by character
    while ((ch = fgetc(fp1)) != EOF) {
        fputc(ch, fp2);
    }

    printf("File copied successfully.\n");
```

```
    fclose(fp1);
    fclose(fp2);

    return 0;
}
```

Sample Output

```
File copied successfully.
```

If **source.txt** contains:

```
Hello World
This is a test file.
```

iv) Write a C program to merge two files into the third file using command-line arguments.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    FILE *fp1, *fp2, *fp3;
    char ch;

    // Check for correct number of arguments
    if (argc != 4) {
        printf("Usage: %s <file1> <file2> <outputfile>\n", argv[0]);
        return 1;
    }

    // Open first file for reading
    fp1 = fopen(argv[1], "r");
    if (fp1 == NULL) {
        printf("Error opening file: %s\n", argv[1]);
        return 1;
    }

    // Open second file for reading
    fp2 = fopen(argv[2], "r");
    if (fp2 == NULL) {
        printf("Error opening file: %s\n", argv[2]);
        fclose(fp1);
        return 1;
    }

    // Open output file for writing
```

```
    fp3 = fopen(argv[3], "w");
    if (fp3 == NULL) {
        printf("Error opening output file: %s\n", argv[3]);
        fclose(fp1);
        fclose(fp2);
        return 1;
    }

    // Copy contents of file1 to output file
    while ((ch = fgetc(fp1)) != EOF) {
        fputc(ch, fp3);
    }

    // Copy contents of file2 to output file
    while ((ch = fgetc(fp2)) != EOF) {
        fputc(ch, fp3);
    }

    printf("Files merged successfully into %s\n", argv[3]);

    fclose(fp1);
    fclose(fp2);
    fclose(fp3);

    return 0;
}
```

⭐ Sample Output

csharp Copy code

```
Files merged successfully into output.txt
```

If:

- **file1.txt** contains → `Hello`
- **file2.txt** contains → `World`

Then **output.txt** will contain:

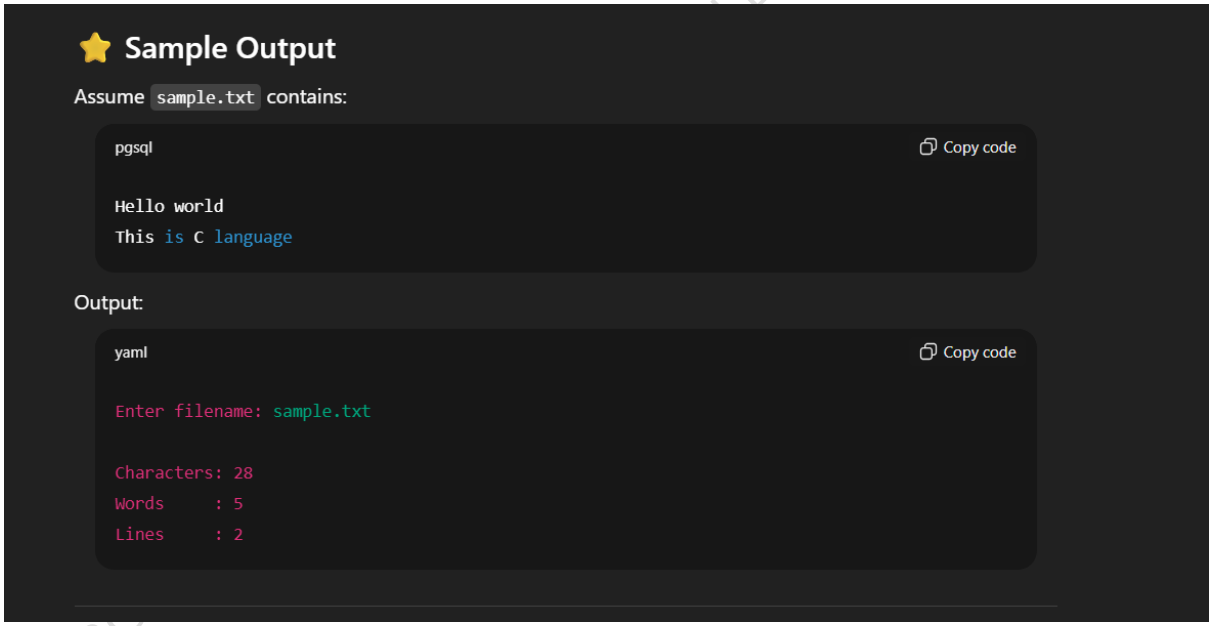
nginx Copy code

```
HelloWorld
```

v) Find no. of lines, words and characters in a file

```
#include <stdio.h>

int main() {
    FILE *fp;
    char filename[50], ch;
    int lines = 0, words = 0, characters = 0;
    int inWord = 0;

    printf("Enter filename: ");
    scanf("%s", filename);

    fp = fopen(filename, "r");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 0;
    }

    while ((ch = fgetc(fp)) != EOF) {
        characters++;

        // Count lines
        if (ch == '\n')
            lines++;

        // Count words
        if (ch == ' ' || ch == '\n' || ch == '\t') {
            inWord = 0;
        } else if (inWord == 0) {
```

```
            inWord = 1;
            words++;
        }
    }

    fclose(fp);

    printf("\nCharacters: %d\n", characters);
    printf("Words     : %d\n", words);
    printf("Lines     : %d\n", lines);

    return 0;
}
```



```c
#include <stdio.h>

int main() {
    FILE *fp;
    char filename[50];
    int n, length;
    char ch;

    printf("Enter filename: ");
    scanf("%s", filename);

    printf("Enter number of characters to print from end: ");
    scanf("%d", &n);

    fp = fopen(filename, "r");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 0;
    }

    // Move to end of file and get length
    fseek(fp, 0, SEEK_END);
    length = ftell(fp);

    // If n is larger than file size, adjust n
    if (n > length)
```

```
    n = length;

    // Move pointer "n" characters back
    fseek(fp, -n, SEEK_END);

    printf("\nLast %d characters:\n", n);

    // Print from that position to end
    while ((ch = fgetc(fp)) != EOF) {
        putchar(ch);
    }

    fclose(fp);
    return 0;
}
```