COMPUTER PROGRAMMING LAB MANUAL

M.JAGADEESH,
ASSOCIATE PROFESSOR,
DEPARTMENT OF CAI,
SJCET.

COMPUTER PROGRAMMING LAB

(Common to All branches of Engineering)

Course Objectives:

The course aims to give students hands – on experience and train them on the concepts of the C-programming language.

Course Outcomes:

CO1: Read, understand, and trace the execution of programs written in C language.

CO2: Select the right control structure for solving the problem.

CO3: Develop C programs which utilize memory efficiently using programming constructs like pointers.

CO4: Develop, Debug and Execute programs to demonstrate the applications of arrays, functions, basic concepts of pointers in C

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

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

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 II WEEK 4

Objective: Explore the full scope of expressions, type-compatibility of variables & constants and operators used in the expression and how operator precedence works. Suggested Experiments/Activities: Tutorial4: Operators and the precedence and as associativity: Lab4: Simple computational problems using the operator' precedence and associativity

i) Evaluate the following expressions. a. A+B*C+(D*E) + F*G

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

c. A+++B---A d.

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

ii) Find the maximum of three numbers using conditional operator

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

WEEK 5

Objective: Explore the full scope of different variants of "if construct" namely if-else, nullelse,

if-else if*-else, switch and nested-if including in what scenario each one of them can be used and how to use them. Explore all relational and logical operators while writing conditionals for "if construct". Suggested Experiments/Activities: Tutorial 5: Branching and logical expressions: Lab 5: Problems involving if-then-else structures.

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

ii) Write a C program to generate electricity bill.

iii) Find the roots of the quadratic equation.

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

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

WEEK 6 Objective:

Explore the full scope of iterative constructs namely while loop, do-while loop and for loop in addition to structured jump constructs like break and continue including when each of these statements is more appropriate to use. Suggested Experiments/Activities:

Tutorial 6: Loops, while and for loops

Lab 6: Iterative problems e.g., the sum of series

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

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

iii) Compute sine and cos series

iv) Checking a number palindrome

v) v) Construct a pyramid of numbers.

UNIT III WEEK 7:

Objective: Explore the full scope of Arrays construct namely defining and initializing 1-D and 2-D and more generically n-D arrays and referencing individual array elements from the defined array. Using integer 1-D arrays, explore search solution linear search. Suggested Experiments/Activities:

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.

ii) Perform linear search on1D array.

iii) The reverse of a 1D integer array

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

V) Eliminate duplicate elements in an array.

WEEK 8:

Objective: Explore the difference between other arrays and character arrays that can be used as Strings by using null character and get comfortable with string by doing experiments that will reverse a string and concatenate two strings. Explore sorting solution bubble sort using integer arrays. Suggested Experiments/Activities: Tutorial 8: 2 D arrays, sorting and Strings.

Lab 8: Matrix problems, String operations, Bubble sort

i) Addition of two matrices

ii) Multiplication two matrices

iii) Sort array elements using bubble sort

iv) Concatenate two strings without built-in functions

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

UNIT IV WEEK 9:

Objective: Explore pointers to manage a dynamic array of integers, including memory allocation & value initialization, resizing changing and reordering the contents of an array and memory de-allocation using malloc (), calloc (), realloc () and free () functions. Gain experience processing command-line arguments received by C Suggested Experiments/Activities: 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()

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

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

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

v) Write a C program to implement realloc()

WEEK 10:

Objective: Experiment with C Structures, Unions, bit fields and self-referential structures (Singly linked lists) and nested structures Suggested Experiments/Activities:

Tutorial 10: Bitfields, Self-Referential Structures,

Linked lists Lab10 : Bitfields, linked lists Read and print a date using dd/mm/yyyy format using bit-fields and differentiate the same without using bit- fields

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

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

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

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

UNIT V WEEK 11:

Objective: Explore the Functions, sub-routines, scope and extent of variables, doing some experiments by parameter passing using call by value. Basic methods of numerical integration Suggested Experiments/Activities: Tutorial 11: Functions, call by value, scope and extent, Lab 11: Simple functions using call by value, solving differential equations using Eulers theorem.

i) Write a C function to calculate NCR value.

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

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

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

WEEK 12: Objective: Explore how recursive solutions can be programmed by writing recursive functions that can be invoked from the main by programming at-least five distinct problems that have naturally recursive solutions

Lab 12: Recursive functions

i) Write a recursive function to generate Fibonacci series.

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

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

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

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

WEEK 13:

Objective: Explore the basic difference between normal and pointer variables, Arithmetic operations using pointers and passing variables to functions using pointers Suggested Experiments/Activities: Tutorial 13: Call by reference, dangling pointers Lab 13: Simple functions using Call by reference, Dangling pointers.

i) Write a C program to swap two numbers using call by reference.
ii) Demonstrate Dangling pointer problem using a C program.
iii) Write a C program to copy one string into another using pointer.
iv) Write a C program to find no of lowercase, uppercase, digits and other characters using pointers.

WEEK14: Objective: To understand data files and file handling with various file I/O functions. Explore the differences between text and binary files. Suggested Experiments/Activities: Tutorial 14: File handling Lab 14: File operations

i) Write a C program to write and read text into a file.
ii) Write a C program to write and read text into a binary file using fread() and fwrite()
iii) Copy the contents of one file to another file.
iv) Write a C program to merge two files into the third file using command-line arguments.
v) Find no. of lines, words and characters in a file
vi) Write a C program to print last n characters of a given file.

WEEK-I

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()

i) Basic Linux Environment:
To get started with writing C programs in a Linux environment, you'll need:
A Linux distribution (e.g., Ubuntu, CentOS, Debian).
A terminal emulator (e.g., GNOME Terminal, Konsole, or xterm).
A C compiler, such as GCC (GNU Compiler Collection), which is usually pre-installed on most Linux distributions.

Writing a Simple C Program:
Let's create a simple "Hello, World!" C program using a text editor:

```
#include <stdio.h>
int main() {
   printf("Hello, World!\n");
   return 0;
}
```

Save this code to a file, for example, **hello.c**.

Text Editors:
Now, let's look at how to use Vi, Vim, and Emacs to edit and compile this C program.
Vi:
Open the terminal and type:

```
Hello.c
```

This opens the Vi editor. To edit the file, press **i**to enter insert mode, make your changes, and then press **Esc**to exit insert mode. To save and exit, type **:wq**and press Enter.
Vim:
Vim is an improved version of Vi. To open the file with Vim, you can use:

```
Vim hello.c
```

To edit, press i, make changes, and exit insert mode with Esc. Save and exit with :wq.
Emacs:
You can open the file in Emacs with:

```
Emacs hello.c
```

In Emacs, you can edit by simply typing. To save, press **Ctrl + x**followed by **Ctrl + s**. To exit, press **Ctrl + x**followed by **Ctrl + c**.
Compiling and Running the C Program:

After saving the C program, compile it using GCC:
`Gcc hello.c –o hello`

This will create an executable named hello. You can run it by typing:

./hello
You should see the output: "Hello, World!"
That's the basic process for creating and editing C programs in a Linux environment using text editors like Vi, Vim, and Emacs. These editors have their own learning curves, so it's beneficial to practice and explore their features to become more proficient with them.

II)

1. Turbo C:

Turbo C was a popular integrated development environment (IDE) for the C programming language, primarily used in the MS-DOS environment. It was developed by Borland and was widely used in the late 1980s and 1990s. Here's how you can get exposure to Turbo C:
Installation: Turbo C is a DOS-based application, and it may not work directly on modern operating systems. You might need to use DOSBox, a DOS emulator, to run Turbo C on modern Windows systems. Alternatively, you can explore online emulators or old machines that support DOS.
Writing and Compiling Code: Turbo C had a simple IDE with a text editor for writing C code. You could write your C programs in the editor, compile them, and run them within the IDE.
Legacy Code: Turbo C uses an outdated C compiler and may not be suitable for modern C programming. It's essential to learn it for historical purposes, but for practical C development, it's recommended to use modern compilers like GCC.

2. GCC (GNU Compiler Collection):

GCC is a widely-used, open-source compiler suite that supports various programming languages, including C. Here's how you can get exposure to GCC for C programming:
Installation: GCC is readily available on most Linux distributions. You can install it using your distribution's package manager. For example, on Ubuntu, you can use:

```
sudo apt-get install gcc
```

Writing and Compiling Code: You can write C code in any text editor or integrated development environment (IDE) of your choice. Save the code with a `.c` extension.
Compiling: To compile a C program with GCC, open the terminal and navigate to the directory containing your C file. Use the following command to compile it:

```
gcc your_program.c -o output_executable
```

Replace `your_program.c`
compiled program.

GCC is a versatile and powerful compiler, suitable for both learning C programming and developing production-quality C code. It's widely used in the Linux and open-source software development communities.

Simple Output Program using `printf()`:

```
 #include <stdio.h>
int main() {
   printf("Hello, World!\n");
   return 0;
}
```

Output: Hello, World

Simple Input Program using scanf():

```c
#include <stdio.h>

int main() {
   int number;
   printf("Enter an integer: ");
   scanf("%d", &number);
   printf("You entered: %d\n", number);
   return 0;
}
```

Output:
Enter an integer: 12
You entered: 12

WEEK-2

Lab 1: Converting algorithms/flow charts into C Source code. Developing the algorithms/flowcharts for the following sample programs

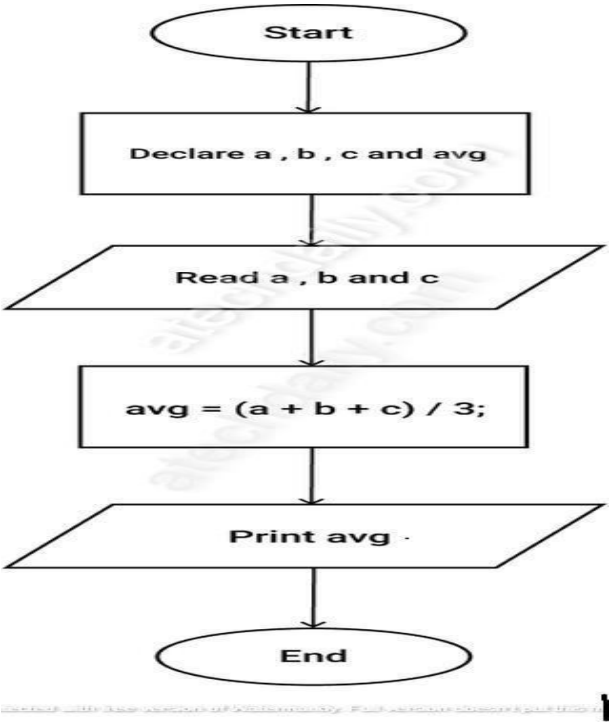
i) Sum and average of 3 numbers
ii) Conversion of Fahrenheit to Celsius and vice versa
iii) Simple interest calculation

`1)Sum and average of 3 numbers

Algorithm to Calculate Sum and Average of Three Numbers:

1. Start
2. Initialize three variables **num1**, **num2**, and **num3** to store the three input numbers.
3. Initialize variables **sum**and **average**to store the sum and average, respectively.
4. Prompt the user to enter the first number (**num1**) and read the input.
5. Prompt the user to enter the second number (**num2**) and read the input.
6. Prompt the user to enter the third number (**num3**) and read the input.
7. Calculate the sum of the three numbers: **sum = num1 + num2 + num3**.
8. Calculate the average: **average = sum / 3.0**(use **3.0**to ensure a floating-point division).
9. Display the sum and average to the user.
10. End.

Flowchart:



```
#include <stdio.h>
int main() {
   // Initialize variables
   double num1, num2, num3, sum, average;
   // Prompt the user to enter the first number
   printf("Enter the first number: ");
   scanf("%lf", &num1);
   // Prompt the user to enter the second number
   printf("Enter the second number: ");
   scanf("%lf", &num2);
   // Prompt the user to enter the third number
   printf("Enter the third number: ");
   scanf("%lf", &num3);
   // Calculate the sum of the three numbers
   sum = num1 + num2 + num3;
   // Calculate the average of the three numbers
   average = sum / 3.0;  // Use 3.0 for floating-point division
   // Display the sum and average
   printf("Sum: %.2lf\n", sum);
   printf("Average: %.2lf\n", average);
   return 0;
}
```

Output: Enter the first number: 12
Enter the second number: 14 Enter
the third number: 15
Sum: 41.00
Average: 13.67

2.Conversion of Fahrenheit to Celsius and vice versa

```
#include <stdio.h>
int main() {
   double fahrenheit, celsius;
   // Prompt the user to enter the temperature in Fahrenheit
   printf("Enter temperature in Fahrenheit: ");
   scanf("%lf", &fahrenheit);

   // Convert Fahrenheit to Celsius
   celsius = (fahrenheit - 32) * 5.0 / 9.0;
   // Display the result
   printf("%.2lf degrees Fahrenheit is equal to %.2lf degrees Celsius.\n", fahrenheit, celsius);
   return 0;
}
```

Output: Enter temperature in Fahrenheit: 4545.00
degrees Fahrenheit is equal to 7.22 degrees Celsius.

celsius, to Fahrenheit:

```
#include <stdio.h>
int main() {
   double celsius, fahrenheit;
   // Prompt the user to enter the temperature in Celsius
   printf("Enter temperature in Celsius: ");
```

```
    scanf("%lf", &celsius);
    // Convert Celsius to Fahrenheit
    fahrenheit = (celsius * 9.0 / 5.0) + 32;
    // Display the result
    printf("%.2lf degrees Celsius is equal to %.2lf degrees Fahrenheit.\n", celsius, fahrenheit);
    return 0;
}
```

Output: Enter temperature in Celsius: 45
45.00 degrees Celsius is equal to 113.00 degrees Fahrenheit.

3. Simple interest calculation in c program

```
#include <stdio.h>
int main() {
    double principal, rate, time, simple_interest;

    // Prompt the user to enter the principal amount
    printf("Enter the principal amount: ");
    scanf("%lf", &principal);

    // Prompt the user to enter the rate of interest printf("Enter
    the rate of interest (in percentage): "); scanf("%lf", &rate);
    // Prompt the user to enter the time period in years
    printf("Enter the time period (in years): ");
    scanf("%lf", &time);
    // Calculate simple interest
    simple_interest = (principal * rate * time) / 100.0;
    // Display the calculated simple interest
    printf("Simple Interest: %.2lf\n", simple_interest);
    return 0;
}
```

Output: Enter the principal amount: 10000
Enter the rate of interest (in percentage): 2
Enter the time period (in years): 2
Simple Interest: 400.00

WEEK 3

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**

1. Finding the square root of a given number

```
#include <stdio.h>
#include <math.h>
int main() {
   double number, square_root;
   // Prompt the user to enter a number
   printf("Enter a number: ");
   scanf("%lf", &number);
   // Calculate the square root using the sqrt() function
   square_root = sqrt(number);
   // Display the square root
   printf("Square root of %.2lf = %.2lf\n", number, square_root);
   return 0;
}
```

Output:

```
Enter a number: 25
Square root of 25.00 = 5.00
```

2. Finding compound interest in c program

Compound Interest (CI)=$P(1+R/n)nt-P$

Where:

CI is the compound interest.

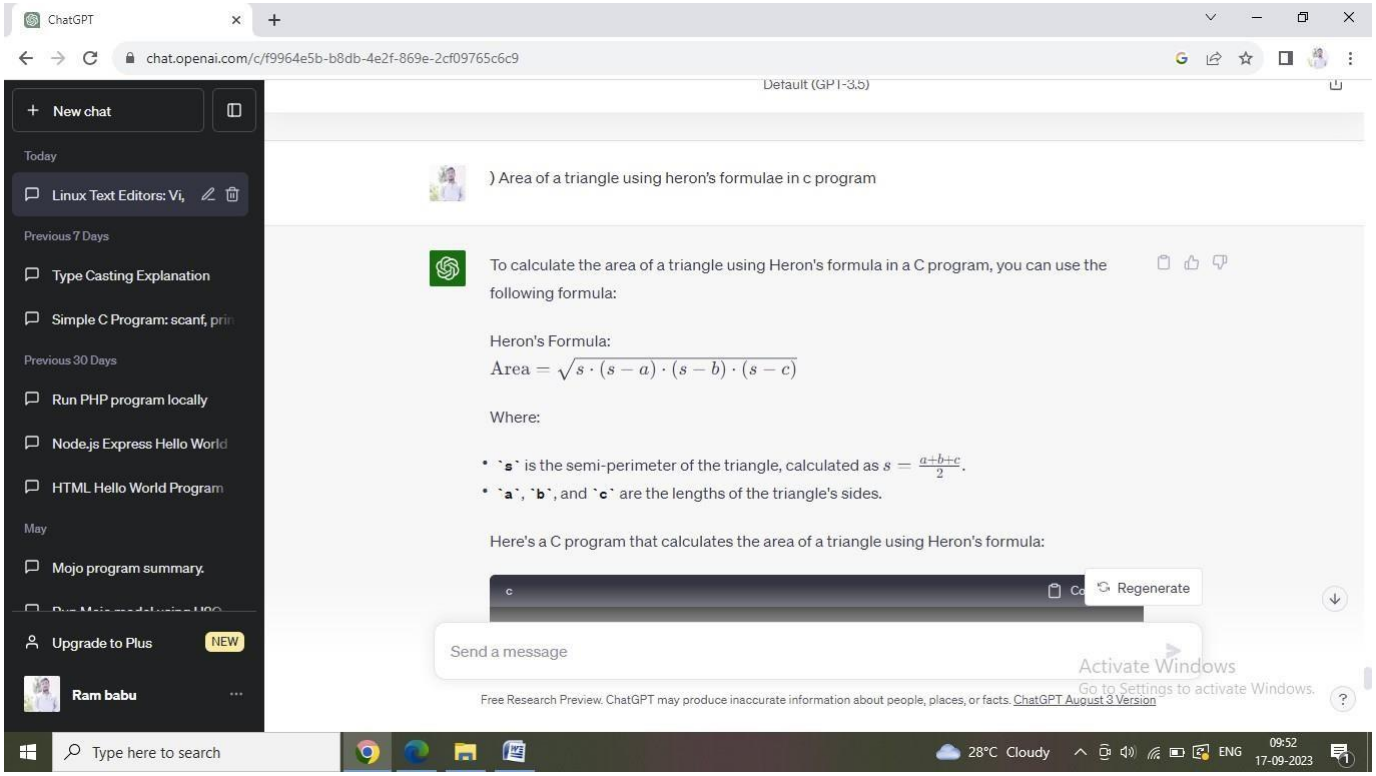
- *P* is the principal amount (the initial amount of money).
- *R* is the annual interest rate (in decimal form).
- *n* is the number of times interest is compounded per year.
- *t* is the time period in years.

```
#include <stdio.h>
#include <math.h>
int main() {
   double principal, rate, time, compound_interest;
   int n;
   // Prompt the user to enter the principal amount
   printf("Enter the principal amount: ");
   scanf("%lf", &principal);
   // Prompt the user to enter the annual interest rate (in percentage)
   printf("Enter the annual interest rate (in percentage): ");
   scanf("%lf", &rate);
   // Prompt the user to enter the number of times interest is compounded per year
   printf("Enter the number of times interest is compounded per year: ");
```

```
    scanf("%d", &n);
    // Prompt the user to enter the time period in years
    printf("Enter the time period (in years): ");
    scanf("%lf", &time);
    // Convert the annual interest rate to decimal form
    rate = rate / 100.0;
    // Calculate compound interest
    compound_interest = principal * pow(1 + rate / n, n * time) - principal;

    // Display the calculated compound interest
    printf("Compound Interest: %.2lf\n", compound_interest);

    return 0;
}
```

Output: Enter the principal amount: 1000
Enter the annual interest rate (in percentage): 2
Enter the number of times interest is compounded per year: 1 Enter
the time period (in years): 1
Compound Interest: 20.00

2.) Area of a triangle using heron's formulae in c program:

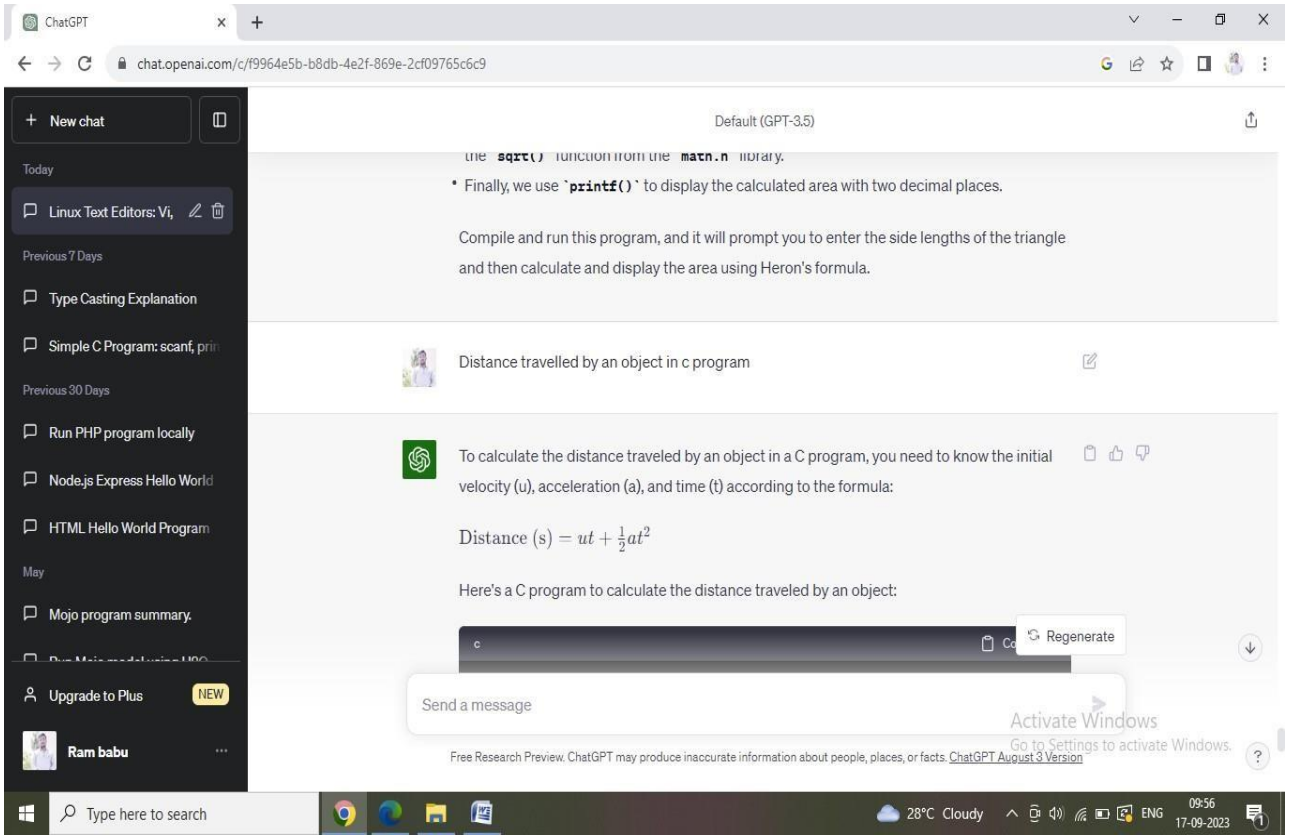



```
#include <stdio.h>
#include <math.h>
int main() {
    double a, b, c, s, area;
    // Prompt the user to enter the lengths of the three sides of the triangle
    printf("Enter the lengths of the three sides of the triangle:\n"); printf("Side
    a: ");
    scanf("%lf", &a);
    printf("Side b: ");
```

```
    scanf("%lf", &b);
    printf("Side c: ");
    scanf("%lf", &c);

    // Calculate the semi-perimeter s
    = (a + b + c) / 2.0;
    // Calculate the area using Heron's formula
    area = sqrt(s * (s - a) * (s - b) * (s - c));
    // Display the calculated area
    printf("The area of the triangle is: %.2lf\n", area);
    return 0;
}
```

Output: Enter the lengths of the three sides of the triangle:
Side a: 5
Side b: 6
Side c: 9
The area of the triangle is: 14.14

4, Distance travelled by an object:




```
#include <stdio.h>
int main() {
    double initial_velocity, acceleration, time, distance;
    // Prompt the user to enter the initial velocity (u) in meters per second (m/s)
    printf("Enter the initial velocity (u) in m/s: ");
    scanf("%lf", &initial_velocity);
```

```
    // Prompt the user to enter the acceleration (a) in meters per second squared (m/s^2)
    printf("Enter the acceleration (a) in m/s^2: ");
    scanf("%lf", &acceleration);
    // Prompt the user to enter the time (t) in seconds (s)
    printf("Enter the time (t) in seconds: ");
    scanf("%lf", &time);
    // Calculate the distance using the formula
    distance = initial_velocity * time + 0.5 * acceleration * time * time;
    // Display the calculated distance
    printf("The distance traveled by the object is %.2lf meters.\n", distance);
    return 0;
}
```

Output: Enter the initial velocity (u) in
m/s: 5 Enter the acceleration (a) in m/s^2:
20
Enter the time (t) in seconds: 120
The distance traveled by the object is 144600.00 meters.

WEEK 4

Lab4: Simple computational problems using the operator' precedence and associativity

i) Evaluate the following expressions.

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

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

c. A+++B---A

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

ii) Find the maximum of three numbers using conditional operator

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

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

```
#include <stdio.h>
int main() {
   // Declare variables
   double A, B, C, D, E,
   F, G; double result;

   // Prompt the user to enter values for A, B, C, D, E, F, and G
   printf("Enter the value of A: ");
   scanf("%lf", &A);
   printf("Enter the value of B: ");
   scanf("%lf", &B);
   printf("Enter the value of C: ");
   scanf("%lf", &C);
   printf("Enter the value of D: ");
   scanf("%lf", &D);
   printf("Enter the value of E: ");
   scanf("%lf", &E);
   printf("Enter the value of F: ");
   scanf("%lf", &F);
   printf("Enter the value of G: ");
  scanf("%lf", &G);
   // Evaluate the expression
   result = A + B * C + (D * E) + F * G;
   // Display the result
   printf("Result of the expression: %.2lf\n", result);
   return 0;
}
```

Output: Enter the value of A: 6 Enter the value of B: 7 Enter the value of C: 5 Enter the value of D: 2 Enter the value of E: 8 Enter the value of F: 3 Enter the value of G: 1

Result of the expression: 60.00

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

To evaluate the expression (/ · − + · /3)($A/B \cdot C - B + A \cdot D/3$) in a C program, you can use the following code:

```
#include <stdio.h>
 int main() {
   double A, B, C, D, result;
   // Prompt the user to enter values for A, B, C, and D
   printf("Enter values for A, B, C, and D:\n");
   printf("A: ");
   scanf("%lf", &A);
   printf("B: ");
   scanf("%lf", &B);
   printf("C: ");
   scanf("%lf", &C);
   printf("D: ");
   scanf("%lf", &D);
   // Evaluate the expression
   result = (A / B * C - B + A * D / 3);
   // Display the result printf("Result: %.2lf\n", result);
   return 0;
}
```

Output: Enter values for A, B, C,
and D: A: 5
B: 6
C: 4
D: 8
Result: 10.67

c. A+++B---A

```
#include <stdio.h>
int main() {
   int A = 5, B = 10;
   printf("Initial values: A = %d, B = %d\n", A, B);
   A++;  // Increment A by 1
   B--;  // Decrement B by 1
   A++;   // Increment A by 1
   printf("After operations: A = %d, B = %d\n", A, B);
   return 0;
}
```

Output: Initial values: A = 5, B = 10
After operations: A = 7, B = 9

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

```
#include <stdio.h>
int main() {
    int i = 5;
    int J;
    J = (i++) + (++i);
    printf("J = %d\n", J);

    return 0;
}
```

Output: J = 12

2.Find the maximum of three numbers using conditional operator

```
#include <stdio.h>
int main() {
  int num1, num2, num3, max;
  // Prompt the user to enter three numbers
  printf("Enter three numbers: ");
  scanf("%d %d %d", &num1, &num2, &num3);
  // Find the maximum using the conditional operator
  max = (num1 > num2) ? ((num1 > num3) ? num1 : num3) : ((num2 > num3) ? num2 : num3);
  // Display the maximum number
  printf("The maximum number is: %d\n", max);
  return 0;
}
```

Out put: Enter three numbers:
10 5
15
The maximum number is: 15

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

```
#include <stdio.h>
int main() {
  int marks[5];
  // Array to store marks for 5 subjects
  int total = 0;
   // Variable to store the total marks
  float average; // Variable to store the average
  // Prompt the user to enter marks for each
  subject printf("Enter marks for 5 subjects:\n");
  for (int i = 0; i < 5; i++) {
    printf("Subject %d: ", i + 1);
    scanf("%d", &marks[i]);
    total += marks[i]; // Add the marks to the total
```

```
    }

    // Calculate the average
    average = (float)total / 5; // Ensure floating-point division
    // Display the total and average marks
    printf("Total marks: %d\n", total);
    printf("Average marks: %.2f\n",
    average);
    return 0;
}
```

Output: Enter marks for 5 subjects:
Subject 1: 50
Subject 2: 60
Subject 3: 70
Subject 4: 55
Subject 5: 80
Total marks: 315
Average marks: 63.00

WEEK 5

Lab 5: Problems involving if-then-else structures.

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

ii) ii) Write a C program to generate electricity bill.

iii) iii) Find the roots of the quadratic equation.

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

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

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

```
#include <stdio.h>
int main() {
   int num1, num2, num3, num4;
   int max, min;
   // Prompt the user to enter four numbers
   printf("Enter four numbers: ");
   scanf("%d %d %d %d", &num1, &num2, &num3, &num4);
   // Assume num1 is both max and min initially
   max = min = num1;
   // Compare num2 with max and
   min if (num2 > max)
      max = num2;
   else if (num2 < min)
      min = num2;
   // Compare num3 with max and
   min if (num3 > max)
      max = num3;
   else if (num3 < min)
      min = num3;
   // Compare num4 with max and
   min if (num4 > max)
      max = num4;
   else if (num4 < min)
      min = num4;
   // Display the maximum and minimum numbers
   printf("Maximum number: %d\n", max);
   printf("Minimum number: %d\n", min);
   return 0;
}
```

Output: Enter four
numbers: 5 6
8
9
Maximum number: 9
Minimum number: 5

2. Write a C program to generate electricity bill.

```
#include <stdio.h>
int main() {
  int consumerNumber;
  float unitsConsumed, ratePerUnit, totalBill;

  // Prompt the user to enter the consumer number
  printf("Enter Consumer Number: ");
  scanf("%d", &consumerNumber);

  // Prompt the user to enter the units consumed
  printf("Enter Units Consumed: ");
  scanf("%f", &unitsConsumed);

  // Prompt the user to enter the rate per
  unit printf("Enter Rate Per Unit: ");
  scanf("%f", &ratePerUnit);

  // Calculate the total bill
  totalBill = unitsConsumed * ratePerUnit;

  // Display the electricity bill
  printf("\nElectricity Bill\n");
  printf("Consumer Number: %d\n", consumerNumber);
  printf("Units Consumed: %.2f\n", unitsConsumed);
  printf("Rate Per Unit: %.2f\n", ratePerUnit);
  printf("Total Bill: Rs. %.2f\n", totalBill);

  return 0;
}
```

Output: Enter Consumer Number:
182 Enter Units Consumed: 70
Enter Rate Per Unit: 2
Electricity Bill
Consumer Number:
182
Units Consumed:
70.00 Rate Per Unit:
2.00 Total Bill: Rs.
140.00

3. Find the roots of the quadratic equation.

```
#include <stdio.h>
#include <math.h>
int main() {
   double a, b, c, discriminant, root1, root2;
   // Prompt the user to enter coefficients a, b, and c
   printf("Enter coefficients a, b, and c: ");
   scanf("%lf %lf %lf", &a, &b, &c);
   // Calculate the discriminant (D = b^2 - 4ac)
    discriminant = b * b - 4 * a * c;
   // Check the value of the discriminant
   if (discriminant > 0) {
      // Two real and distinct roots
      root1 = (-b + sqrt(discriminant)) / (2 * a);
      root2 = (-b - sqrt(discriminant)) / (2 * a);
      printf("Root 1 = %.2lf\n", root1);
      printf("Root 2 = %.2lf\n", root2);
   } else if (discriminant == 0) {
      // One real root (both roots are the
      same) root1 = -b / (2 * a);
      printf("Root 1 = Root 2 = %.2lf\n", root1);
   } else {
      // Complex roots (no real roots)
      double realPart = -b / (2 * a);
      double imaginaryPart = sqrt(-discriminant) / (2 * a);
      printf("Root 1 = %.2lf + %.2lfi\n", realPart, imaginaryPart);
      printf("Root 2 = %.2lf - %.2lfi\n", realPart, imaginaryPart);
   }
   return 0;
}
```

OUTPUT: Enter coefficients a, b,
and c: 5 6
7
Root 1 = -0.60 + 1.02i
Root 2 = -0.60 - 1.02i

4. Write a C program to simulate a calculator using switch case

```
#include <stdio.h>
int main() {
   char operator;
   double num1, num2, result;
   // Prompt the user to enter an operator (+, -, *, /)
   printf("Enter an operator (+, -, *, /): ");
   scanf(" %c", &operator);
```

```
    // Prompt the user to enter two numbers
    printf("Enter two numbers: ");
    scanf("%lf %lf", &num1, &num2);
    switch (operator) {
      case '+':
        result = num1 + num2;
        printf("Result: %.2lf\n",
        result); break;
      case '-':
        result = num1 - num2;
        printf("Result: %.2lf\n",
        result); break;
      case '*':
        result = num1 * num2;
        printf("Result: %.2lf\n",
        result); break;
      case '/':
        if (num2 != 0) {
          result = num1 / num2;
          printf("Result: %.2lf\n", result);
    } else {
          printf("Error: Division by zero is not allowed.\n");
        }
        break
      default:
        printf("Error: Invalid operator.\n");
    }

    return 0;
}
```

Output: Enter an operator (+, -, *, /): *
Enter two numbers: 20
5
Result: 100.00

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

If a year is divisible by 4, it is a leap year.
However, if that year is divisible by 100, it is not a leap year. But if it is also divisible by 400, then it is a leap year.

```
#include <stdio.h>
int main()
{
  int year;
```

```
printf("Enter a year:
"); scanf("%d",
&year);
if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
 {
 printf("%d is a leap year.\n", year);
 }
Else
{
 printf("%d is not a leap year.\n", year);
  }

 return 0;
          }
```

Out put: Enter a year: 2024
2024 is a leap year.

WEEK 6

Objective: Explore the full scope of iterative constructs namely while loop, do-while loop and

Suggested Experiments/Activities:

Tutorial 6: Loops, while and for loops

Lab 6: Iterative problems e.g., the sum of series

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

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

iii) Compute sine and cos series

iv) Checking a number palindrome

v) Construct a pyramid of numbers.

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

Source code:

```
#include<stdio.h>
int main()
{
 int i,fact=1,number;
 printf("Enter a number: ");
 scanf("%d",&number);
    for(i=1;i<=number;i++){
      fact=fact*i;
  }
  printf("Factorial of %d is: %d",number,fact);
return 0;
}
```

Out put: Enter a number: 5 Factorial of
5 is: 120

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

Source code:

```
#include <stdio.h>
int main() {
 int n, i, flag = 0;
 printf("Enter a positive integer: ");
 scanf("%d", &n);
 // 0 and 1 are not prime numbers
 // change flag to 1 for non-prime number
 if (n == 0 || n == 1)
  flag = 1;
  for (i = 2; i <= n / 2; ++i) {

  // if n is divisible by i, then n is not prime
  // change flag to 1 for non-prime
  number if (n % i == 0) {
    flag = 1;
    break;
```

```
  }
 }

 // flag is 0 for prime
 numbers if (flag == 0)
  printf("%d is a prime number.",
 n); else
  printf("%d is not a prime number.", n);

 return 0;
}
```

Output:

Enter a positive integer: 29

29 is a prime number.

iii) Compute sine and cos series

```
 //cosine values
 #include<stdio.h>
 #include<math.h>
 #define PI 3.1416
 #define MAX 150
 main (){
    int   angle;
    float    x,y;
    angle =0;
    printf("Angle cos(angle)");
    while(angle <= MAX){ x =(PI/MAX)*angle;
       y = cos(x);
       printf("%15d %13.4f
 ", angle, y);
       angle = angle +10;
    }
 }
```

Output

When the above program is executed, it produces the following output −

```
Angle cos(angle)
0 1.0000
10 0.9781
20 0.9135
30 0.8090
40 0.6691
50 0.5000
60 0.3090
70 0.1045
80 -0.1045
90 -0.3090
100 -0.5000
110 -0.6691
120 -0.8090
```

130 -0.9135
140 -0.9781
150 -1.0000

Example

Following is the C program to find the sine values –

```
//sine values #include<stdio.h> #include<math.h> #define PI 3.1416
        #define MAX 150 main (){
        int angle;  float x,y;
        angle =0;
        printf("Angle sin(angle) ");
   while(angle  <=  MAX)
   {
   x =(PI/MAX)*angle;
   y = sin(x); printf("%15d%13.4f", angle, y);
        angle = angle +10;
        }
}
```

Output

When the above program is executed, it produces the following output −

Angle sin(angle)

0 0.0000
10 0.2079
20 0.4067
30 0.5878
40 0.7431
50 0.8660
60 0.9511
70 0.9945
80 0.9945
90 0.9511
100 0.8660
110 0.7431
120 0.5878
130 0.4067
140 0.2079
150 -0.0000

iv) Checking a number palindrome

Palindrome number in c: A **palindrome number** is *a number that is same after reverse*. For example 121, 34543, 343, 131, 48984 are the palindrome numbers.

```
#include<stdio.h>
int main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)     enter the number=5621
{
  not palindrome   number
r=n%10;
sum=(sum*10)+r;
n=n/10;
}
if(temp==sum)
printf("palindrome number ");
else
printf("not palindrome");

return 0;
}
```

Output:

```
        enter the number=151
palindrome  number
```

v) Construct a pyramid of numbers.

Source code:

```
#include <stdio.h>
int main()
{
        int rows;
        printf("Number of rows: ");
        scanf("%d", &rows);
        // first loop to print all
        rows for (int i = 1; i <=
        rows; i++) {

            // inner loop 1 to print white spaces
            for (int j = 1; j <= 2 * (rows - i); j++) {
        printf(" ");
            }
            // inner loop 2 to print numbers
            for (int k = 1; k < 2 * i; k++) {

        printf("%d ", i);
            }
```

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

Output :

```
Number of rws: 5

          1
        2  2  2
      3  3  3  3  3
    4 4  4  4  4  4 4
  5 5 5  5  5  5  5 5 5
```

WEEK 7:

Suggested Experiments/Activities:
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.
ii) Perform linear search on1D array.
iii) The reverse of a 1D integer array
iv) Find 2's complement of the given binary number.
v) Eliminate duplicate elements in an array.

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

Program:

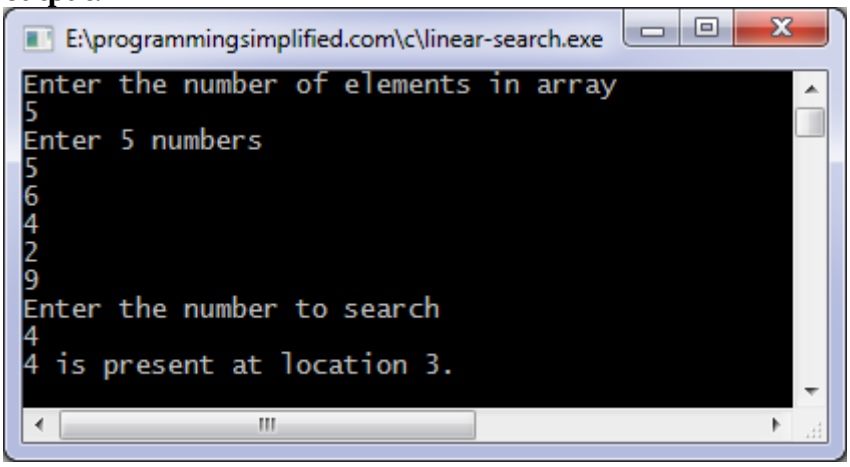
```
 #include <conio.h>
  int main()
 {
    int a[1000],i,n,min,max;

    printf("Enter size of the array : ");
    scanf("%d",&n);

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

    min=max=a[0];
    for(i=1; i<n; i++)
    {
        if(min>a[i])
                    min=a[i];
                     if(max<a[i])
                      max=a[i];
    }
     printf("minimum of array is : %d",min);
         printf("\nmaximum of array is : %d",max);

   return 0;
 }
```

Out put: Enter size of array:5
Enter the elements in array:1
2
3
4
5
Minimum elements of array:1
Maximum elements of array:5

ii) Perform linear search on1D array.

```
#include <stdio.h>
int main()
{
  int array[100], search, c, n;
  printf("Enter number of elements in array\n");
  scanf("%d", &n);
  printf("Enter %d integer(s)\n", n);
  for (c = 0; c < n; c++)
  scanf("%d", &array[c]);
  printf("Enter a number to search\n");
  scanf("%d", &search);
  for (c = 0; c < n; c++)
  {
    if (array[c] == search)     /* If required element is found */
    {
      printf("%d is present at location %d.\n", search, c+1);
      break;
    }
  }
  if (c == n)
    printf("%d isn't present in the array.\n", search);
  return 0;
}
```

output:



iii) The reverse of a 1D integer array

```
#include<stdio.h>
int main()
{
    int n, arr[n], i;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    printf("Enter the elements: ");
    for(i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    int rev[n], j = 0;
    for(i = n-1; i >= 0; i--)
```

```
    {
        rev[j] = arr[i];
        j++;
    }
    printf("The Reversed array: ");
    for(i = 0; i < n; i++)
    {
        printf("%d ", rev[i]);
    }
}
```

Output:

```
Enter the size of the array: 5 Enter the
elements: 1 2 3 4 5
The Reversed array: 5 4 3 2 1
```

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

```
/**
 * C program to find twos complement of a binary number
 */

#include <stdio.h>
#define SIZE 8
int main()
{
    char binary[SIZE + 1], onesComp[SIZE + 1], twosComp[SIZE + 1];
    int i, carry=1;

    printf("Enter %d bit binary value: ", SIZE);

    /* Input 8-bit binary string */
    gets(binary);

    /* Find ones complement of the binary number */
    for(i=0; i<SIZE; i++)
    {
        if(binary[i] == '1')
        {
            onesComp[i] = '0';
        }
        else if(binary[i] == '0')
        {
            onesComp[i] = '1';
        }
    }
    onesComp[SIZE] = '\0';

    /*
     * Add 1 to the ones complement
     */
    for(i=SIZE-1; i>=0; i--)
    {
        if(onesComp[i] == '1' && carry == 1)
        {
```

```
            twosComp[i] = '0';
        }
        else if(onesComp[i] == '0' && carry == 1)
        {
            twosComp[i] = '1';
            carry = 0;
        }
        else
        {
            twosComp[i] = onesComp[i];
        }
    }
    twosComp[SIZE] = '\0';

    printf("Original binary = %s\n", binary);
    printf("Ones complement = %s\n", onesComp);
    printf("Twos complement = %s\n", twosComp);

    return 0;
}
```

Output: Enter 8 bit binary value: 01101100

```
Original binary  = 01101100
Ones complement  = 10010011
Twos complement  = 10010100
```

v) Eliminate duplicate elements in an array.

```
/* program to delete the duplicate elements from sorted array in C. */
#include <stdio.h>

int duplicate_element ( int arr[], int num)
{
    // check num is equal to 0 and num == 1
    if (num == 0 || num == 1)
        return num;
    // create temp array to store same number
    int temp [num];
    // declare variable
    int i, j = 0;

    // use for loop to check duplicate element
    for (i = 0; i < num - 1; i++)
    {
        // check the element of i is not equal to (i + 1) next element
        if (arr [i] != arr[i + 1])
            temp[j++] = arr[i];
    }
    temp[j++] = arr[ num - 1];
    // check the original array's elements with temporary array's elements
    for (i = 0; i < j; i++)
```

```
    {
        arr[i] = temp[i];
        }

    return j;
}

int main ()
{
    int num;
    printf (" Define the no. of elements of the array: ");
    scanf (" %d", &num);
    int arr[num], i;
    printf (" Enter the elements: ");
    // use loop to read elements one by one
    for ( i = 0; i < num; i++)
    {
        scanf (" %d", &arr[i]);
    }
    printf (" \n Elements before removing duplicates: ");
    for ( i = 0; i < num; i++)
    {
        printf (" %d", arr[i]);
    }
    num = duplicate_element (arr, num);
    // print array after removing duplicates elements
    printf (" \n Display array's elements after removing duplicates: ");
    for ( i = 0; i < num; i++)
    {
        printf (" %d", arr[i]);
        }
    return 0;

}
```

When we execute the above program in C compiler, it produces the given below output in the console screen.

```
Define the no. of elements of the array: 10 Enter the
  elements: 5
6
6
7
8
8
9
10
11
11

  Elements before removing duplicates:   5 6 6 7 8 8 9 10 11 11
  Display array's elements after removing duplicates:   5 6 7 8 9 10 11
```

WEEK 8:

2 D arrays, sorting and Strings.
Lab 8: Matrix problems, String operations, Bubble sort
i) Addition of two matrices
ii) Multiplication two matrices
iii) Sort array elements using bubble sort
iv) Concatenate two strings without built-in functions
v) Reverse a string using built-in and without built-in string functions

i) Addition of two matrices

```
#include <stdio.h>
int main() {
    // Declare a 2D array with 3 rows and 4 columns
    int matrix[3][4];

    // Assign values to the matrix
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            matrix[i][j] = i * 4 + j;
        }
    }

    // Print the matrix
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output

```
0 1 2 3
4 5 6 7
8 9 10 11
```

ii) Multiplication two matrices

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
system("cls");
printf("enter the number of row=");
scanf("%d",&r);
printf("enter the number of column=");
scanf("%d",&c);
printf("enter the first matrix element=\n");
```

```
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("enter the second matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&b[i][j]);
}
}

printf("multiply of the matrix=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
mul[i][j]=0;
for(k=0;k<c;k++)
{
mul[i][j]+=a[i][k]*b[k][j];
}
}
}
//for printing result
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",mul[i][j]);
}
printf("\n");
}
return 0;
}
```

Output:

```
enter the number of row=3
enter the number of column=3
enter the first matrix element=
1 1 1
2 2 2
3 3 3
enter the second matrix element=
1 1 1
2 2 2
3 3 3
multiply of the matrix=
6 6 6
12 12 12
18 18 18
```

iii) Sort array elements using bubble sort in c program.

```
#include <stdio.h>
void bubble_sort(int arr[], int n) {
  int i, j;
  for (i = 0; i < n - 1; i++) {
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        int temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}
int main() {
  int arr[] = {64, 34, 25, 12, 22, 11, 90};
  int n = sizeof(arr) / sizeof(arr[0]);
  bubble_sort(arr,  n);
  printf("Sorted array: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  return 0;
}
```

Output: Sorted array: 11 12 22 25 34 64 90

iv) Concatenate two strings without built-in functions in c program.

```
// C Program to concatenate two
// strings without using strcat
#include <stdio.h>

int main()
{
        // Get the two Strings to be concatenated
        char str1[100] = "Geeks", str2[100] = "World";
        // Declare a new Strings
        // to store the concatenated String
        char str3[100];
        int i = 0, j = 0;
        printf("\nFirst string: %s", str1);
        printf("\nSecond string: %s", str2);
        // Insert the first string
        // in the new string
        while (str1[i] != '\0') {
                str3[j] = str1[i];
                i++;
                j++;
        }
```

```
        // Insert the second string
        // in the new string
        i = 0;
        while (str2[i] != '\0') {
                str3[j] = str2[i];
                i++;
                j++;
        }
        str3[j] = '\0';

        // Print the concatenated string
        printf("\nConcatenated string: %s", str3);

        return 0;
}
```

Output

First string: Good Moring Second

string: Friends

Concatenated string: Good Moring Friends

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

```
#include<stdio.h>
main (){
  char a[50];
  clrscr();
  printf (“enter a string”);
  gets (a);
  strrev (a);
  printf(“reversed string=%s”,a)
  getch ();
}
```

Output

enter a string :Hello reversed
string = olleH

WEEK 9

Lab 9: Pointers and structures, memory dereference.

i) Write a C program to find the sum of a 1D array using malloc()
ii) Write a C program to find the total, average of n students using structures
iii) Enter n students data using calloc() and display failed students list
iv) Read student name and marks from the command line and display the student details along with the total.
v) Write a C program to implement realloc()

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

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

int main()
{
    int* ptr; //declaration of integer pointer
    int limit; //to store array limit
    int i; //loop counter
    int sum; //to store sum of all elements

    printf("Enter limit of the array: ");
    scanf("%d", &limit);

    //declare memory dynamically
    ptr = (int*)malloc(limit * sizeof(int));

    //read array elements
    for (i = 0; i < limit; i++) {
        printf("Enter element %02d: ", i + 1);
        scanf("%d", (ptr + i));
    }

    //print array elements
    printf("\nEntered array elements are:\n");
    for (i = 0; i < limit; i++) {
        printf("%d\n", *(ptr + i));
    }

    //calculate sum of all elements
    sum = 0; //assign 0 to replace garbage value
    for (i = 0; i < limit; i++) {
        sum += *(ptr + i);
    }
    printf("Sum of array elements is: %d\n", sum);

    //free memory
    free(ptr); //hey, don't forget to free dynamically allocated memory.

    return 0;
}
```

Output:

```
Enter limit of the array: 5
Enter element 01: 100
Enter element 02: 200
Enter element 03: 300
Enter element 04: 400
Enter element 05: 500

Entered array elements are:
100
200
300
400
500
Sum of array elements is: 1500
```

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

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

structstudent
{
        int rl;
        char nm[20];
        int m1;
        int m2;
        int m3;
        int t;
        float per;
};
voidmain()
{
        structstudenta;
        clrscr();
        printf(" Enter RollNo, Name amd three sub marks\n");
        scanf("%d%s%d%d%d", &a.rl, &a.nm, &a.m1, &a.m2, &a.m3);
        a.t = a.m1 + a.m2 + a.m3;
        a.per = a.t / 3.0;
        printf("rollno=%d\n", a.rl);
        printf("Name=%sk\n", a.nm);
        printf("m1=%d\n", a.m1);
        printf("m2=%d\n", a.m2);
        printf("m3=%d\n", a.m3);
        printf("total=%d\n", a.t);
        printf("per=%f\n", a.per);
        getch();

 }
```

Out put:

```
Enter RollNo, Name and three sub marks
12 rama 304050

rollno=12
Name=rama
m1=30
m2=40
m3=50
total=120
per=40.000000
```

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

```
#include <stdio.h>
#include <stdlib.h>
// Structure to store student data
struct Student {
    char name[50];
    int marks;
};
int main() {
    int n;
    // Prompt the user to enter the number of students
    printf("Enter the number of students: ");
    scanf("%d", &n);
    // Allocate memory for an array of n Student structures
    struct Student *students = (struct Student *)calloc(n, sizeof(struct Student));

    // Check if memory allocation was successful
    if (students == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
    // Input data for each student
    for (int i = 0; i < n; i++) {
        printf("Enter name for student %d: ", i + 1);
        scanf("%s", students[i].name);
        printf("Enter marks for student %d: ", i + 1);
        scanf("%d", &students[i].marks);
    }

    // Display the list of failed students (marks < 40)
    printf("List of failed students:\n");
    for (int i = 0; i < n; i++) {
        if (students[i].marks < 40) {
                printf("Name: %s, Marks: %d\n", students[i].name, students[i].marks);
        }
    }
    // Free the allocated memory
    free(students);
    return 0;
}
```

Output:

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

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

voidmain()
{
int rl,phy,che,ca,total;
float per;
char nm[20],div[10];
printf("Input the Roll Number of the student :");
scanf("%d",&rl);
printf("Input the Name of the Student :");
scanf("%s",nm);
printf("Input the marks of Physics, Chemistry and Computer Application : ");
scanf("%d%d%d",&phy,&che,&ca);
    total = phy+che+ca;
   per = total/3.0;
if(per>=60)
        strcpy(div,"First");
else
if(per<60&&per>=48)
strcpy(div,"Second"); else
if(per<48&&per>=36)
strcpy(div,"Pass");
else
strcpy(div,"Fail");
printf("\nRoll No : %d\nName of Student : %s\n",rl,nm);
printf("Marks in Physics : %d\nMarks in Chemistry : %d\nMarks in Computer Application :
%d\n",phy,che,ca);
printf("Total Marks = %d\nPercentage = %5.2f\nDivision = %s\n",total,per,div);
}
```

Sample Output:

Input the Roll Number of the student :784
Input the Name of the Student :James
Input the marks of Physics, Chemistry and Computer Application : 70 80 90

Roll No : 784
Name of Student : James
Marks in Physics : 70
Marks in Chemistry : 80
Marks in Computer Application : 90
Total Marks = 240
Percentage = 80.00
Division = First

v) Write a C program to implement realloc()

```
#include<stdio.h>
//To use realloc in our program
#include<stdlib.h>
intmain()
{
char *ptr;
    ptr = NULL;

/*since the ptr is NULL,it will act like malloc function*/
    ptr = realloc(ptr,10);
if(ptr != NULL)
        printf("Memory created successfully\n");
return0;
}
```

Output: Memory created successfully

WEEK 10:

Lab10 : Bitfields, linked lists

Read and print a date using dd/mm/yyyy format using bit-fields and differentiate the same without using bit- fields

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

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

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

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

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

```
#include <stdio.h>
struct node {
        int data;
        struct node* prev_link;
        struct node* next_link;
};
int main()
{
        struct node ob1; // Node1
        // Initialization
        ob1.prev_link = NULL;
        ob1.next_link = NULL;
        ob1.data = 10;
        struct node ob2; // Node2
        // Initialization
        ob2.prev_link = NULL;
        ob2.next_link = NULL;
        ob2.data = 20;
        struct node ob3; // Node3
        // Initialization
        ob3.prev_link = NULL;
        ob3.next_link = NULL;
        ob3.data = 30;
        // Forward links
        ob1.next_link = &ob2;
        ob2.next_link = &ob3;
        // Backward links
        ob2.prev_link = &ob1;
        ob3.prev_link = &ob2;
        // Accessing data of ob1, ob2 and ob3 by ob1
        printf("%d\t", ob1.data);
        printf("%d\t", ob1.next_link->data);
        printf("%d\n", ob1.next_link->next_link->data);
        // Accessing data of ob1, ob2 and ob3 by ob2
        printf("%d\t", ob2.prev_link->data);
        printf("%d\t", ob2.data);
        printf("%d\n", ob2.next_link->data);
        // Accessing data of ob1, ob2 and ob3 by ob3
        printf("%d\t", ob3.prev_link->prev_link->data);
```

```
        printf("%d\t", ob3.prev_link->data);
        printf("%d", ob3.data);
        return 0;
        }
```

Output:

```
10      20      30
10      20      30
10      20      30
```

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

```
// C program to illustrate differences
// between structure and Union
#include <stdio.h>
#include <string.h>
// declaring structure
struct struct_example {
        int integer;
        float decimal;
        char name[20];
};
// declaring union
union union_example {
        int integer;
        float decimal;
        char name[20];
};
void main()
{
        // creating variable for structure and initializing values difference six
        struct struct_example s = { 18, 38, "geeksforgeeks" };
        // creating variable for union and initializing values
        union union_example u = { 18, 38, "geeksforgeeks" };

        printf("structure data:\n integer: %d\n" "decimal: %.2f\n name: %s\n",
                s.integer, s.decimal, s.name);
        printf("\nunion data:\n integer: %d\n" "decimal: %.2f\n name: %s\n",
                u.integer, u.decimal, u.name);
        // difference two and three
        printf("\nsizeof structure : %d\n", sizeof(s));
        printf("sizeof union : %d\n", sizeof(u));

        // difference five
        printf("\n Accessing all members at a time:");
        s.integer = 183;
        s.decimal = 90;
        strcpy(s.name, "geeksforgeeks");
        printf("structure data:\n integer: %d\n " "decimal: %.2f\n name: %s\n", s.integer,
                s.decimal, s.name);
        u.integer = 183;
        u.decimal = 90;
```

```
        strcpy(u.name, "geeksforgeeks");

        printf("\nunion data:\n integer: %d\n " "decimal: %.2f\n name: %s\n", u.integer,
                u.decimal, u.name);
        printf("\n Accessing one member at time:");
        printf("\nstructure data:");
        s.integer = 240;
        printf("\ninteger: %d", s.integer);
        s.decimal = 120;
        printf("\ndecimal: %f", s.decimal);
        strcpy(s.name, "C programming");
        printf("\nname: %s\n", s.name);
        printf("\n union data:");
        u.integer = 240;
        printf("\ninteger: %d", u.integer);
        u.decimal = 120;
        printf("\ndecimal: %f", u.decimal);
        strcpy(u.name, "C programming");
        printf("\nname: %s\n", u.name);
        // difference four
        printf("\nAltering a member value:\n");
        s.integer = 1218;
        printf("structure data:\n integer: %d\n " " decimal: %.2f\n name: %s\n", s.integer, s.decimal,
                s.name);
        u.integer = 1218;
        printf("union data:\n integer: %d\n decimal: %.2f\n name: %s\n", u.integer, u.decimal, u.name);
}
```

Output:

```
structure data:
  integer: 18
decimal: 38.00
  name: geeksforgeeks
union data:
  integer: 18
decimal: 0.00 name:
sizeof structure : 28 sizeof
union : 20
  Accessing all members at a time:structure data: integer: 183
  decimal: 90.00 name:
  geeksforgeeks
union data:
  integer: 1801807207
  decimal:  27732287172115950725811404 8.00
  name: geeksforgeeks
  Accessing one member at time: structure
data:
integer: 240
```

decimal: 120.000000 name:
C programming
union data:
integer: 240
decimal: 120.000000 name:
C programming
Altering a member value: structure data:
integer: 1218
decimal: 120.00 name: C
programming union data:
integer: 1218
decimal: 0.00
name:

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

```
#include <stdio.h>
#define INT_BITS 32

/*Function to left rotate n by d bits*/
int leftRotate(int n, unsigned int d)
{
        /* In n<<d, last d bits are 0. To put first 3 bits of n
        at last, do bitwise or of n<<d with n >>(INT_BITS -
        d) */
        return (n << d) | (n >> (INT_BITS - d));
}

/*Function to right rotate n by d bits*/
int rightRotate(int n, unsigned int d)
{
        /* In n>>d, first d bits are 0. To put last 3 bits of at
                        first, do bitwise or of n>>d with n <<(INT_BITS
        - d) */
        return (n >> d) | (n << (INT_BITS - d));
}

/* Driver program to test above functions */
void main()
{
        int n = 16;
        int d = 2;
        printf("Left Rotation of %d by %d is ", n, d);
        printf("%d", leftRotate(n, d));
        printf(" Right Rotation of %d by %d is ", n, d);
        printf("%d", rightRotate(n, d));
}
```

Output:

Left Rotation of 16 by 2 is 64 Right Rotation of 16 by 2 is 4

Time Complexity: O(1)
Auxiliary Space: O(1)

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

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

// Define a structure
struct Student {
    char name[50];
    int roll_number;
};

int main() {
    // Declare two structure variables
    struct Student student1;
    struct Student student2;

    // Initialize the values in student1
    strcpy(student1.name, "John");
    student1.roll_number = 101;
    // Copy the values from student1 to student2
    student2 = student1;
    // Display the contents of both structure variables
    printf("Student 1:\n");
    printf("Name: %s\n", student1.name);
    printf("Roll Number: %d\n", student1.roll_number);
    printf("\nStudent 2 (copied from Student 1):\n");
    printf("Name: %s\n", student2.name);
    printf("Roll Number: %d\n", student2.roll_number);
    return 0;
}
```

Output: Student 1:
Name: John
Roll Number: 101

Student 2 (copied from Student 1):
Name: John
Roll Number: 101

WEEK 11:

Lab 11: Simple functions using call by value, solving differential equations using Eulers theorem.

i) Write a C function to calculate NCR value.
ii) Write a C function to find the length of a string.
iii) Write a C function to transpose of a matrix.
iv) Write a C function to demonstrate numerical integration of differential equations using Euler's method

i) Write a C function to calculate NCR value.

```
#include <stdio.h>
// Function to calculate the factorial of a number
unsigned long long factorial(int num) {
    unsigned long long result = 1;
    for (int i = 1; i <= num; i++) {
        result *= i;
    }
    return result;
}
// Function to calculate N choose R (NCR)
unsigned long long nCr(int n, int r) {
    if (n < r || n < 0 || r < 0) {
        return 0; // Invalid input
    }
    return factorial(n) / (factorial(r) * factorial(n - r));
}

int main() {
    int n, r;
    // Input values for N and R
    printf("Enter the value of N: ");
    scanf("%d", &n);
    printf("Enter the value of R: ");
    scanf("%d", &r);
    // Calculate and display NCR
    unsigned long long result = nCr(n, r);
    printf("%dC%d = %llu\n", n, r, result);
    return 0;
}
```

Output: Enter the value of N: 10
Enter the value of R: 5
10C5 = 252

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

```
#include <stdio.h> int main() {
  char s[] = "Programming is fun";
  int i;

  [i] != '\0'; ++i);
  printf("Length of the string: %d", i);
  return 0;
}
```

Output:

Length of the string: 18

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

```
#include<stdio.h>
int main(){
    int m, n;
    printf("Enter the number of rows: ");
    canf("%d", &m);
    printf("Enter the number of columns: ");
    scanf("%d", &n);
    int matrix[10^5][10^5];
    printf("Enter the elements of the matrix:\n");
    for(int i=0; i<m; i++){
        for(int j=0; j<n; j++){
            scanf("%d", &matrix[i][j]);
        }
    }
    for(int i=0; i<m; i++){
        for(int j=0; j<n; j++){
            int temp = matrix[i][j];
            matrix[i][j] = matrix[j][i];
            matrix[j][i] = temp;
        }
    }
    printf("The transposed matrix is:\n");
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:

Enter the number of rows: 3 Enter the number of columns: 2 Enter the elements of the matrix:
1 2
2 3
3 4
The transposed matrix is:
1 2 3
2 3 4

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

```
#include<stdio.h>
#include<conio.h>
#define f(x,y) x+y
int main()
{
float x0, y0, xn, h, yn, slope;
int i, n;
clrscr();
printf("Enter Initial Condition\n");
printf("x0 = ");
scanf("%f",&x0);
printf("y0 = ");
scanf("%f",&y0);
printf("Enter calculation point xn = ");
scanf("%f",&xn);
printf("Enter number of steps: ");
scanf("%d",&n);
/* Calculating step size (h) */
 h =(xn-x0)/n;
/* Euler's Method */
printf("\nx0\ty0\tslope\tyn\n");
  printf("\n");
for(i=0; i < n; i++)
  {
  slope =f(x0, y0);
  yn = y0 + h * slope;
printf("%.4f\t%.4f\t%0.4f\t%.4f\n",x0,y0,slope,yn);
  y0 = yn;
    x0 = x0+h;
}
/* Displaying result */
printf("\nValue of y at x = %0.2f is %0.3f",xn, yn);
getch();
return0;
}
```

Euler's Method C Program Output

```
Enter Initial Condition
x0 = 0
y0 = 1
Enter calculation point xn = 1
Enter number of steps: 10
```

x0	y0	slope	yn
0.0000	1.0000	1.0000	1.1000
0.1000	1.1000	1.2000	1.2200
0.2000	1.2200	1.4200	1.3620
0.3000	1.3620	1.6620	1.5282
0.4000	1.5282	1.9282	1.7210
0.5000	1.7210	2.2210	1.9431
0.6000	1.9431	2.5431	2.1974
0.7000	2.1974	2.8974	2.4872
0.8000	2.4872	3.2872	2.8159
0.9000	2.8159	3.7159	3.1875

Value of y at x = 1.00 is 3.187

WEEK 12:

Tutorial 12: Recursion, the structure of recursive calls
Lab 12: Recursive functions
i) Write a recursive function to generate Fibonacci series.
ii) Write a recursive function to find the lcm of two numbers.
iii) Write a recursive function to find the factorial of a number.
iv) Write a C Program to implement Ackermann function using recursion.
v) Write a recursive function to find the sum of series.

i) Write a recursive function to generate Fibonacci series.

```
#include<stdio.h>
void printFibonacci(int n){
    static int n1=0,n2=1,n3;
    if(n>0){
        n3 = n1 + n2;
        n1 = n2;
        n2 = n3;
        printf("%d ",n3);
        printFibonacci(n-1);
    }
}
int main(){
    int n;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Fibonacci Series: ");
    printf("%d %d ",0,1);
    printFibonacci(n-2);//n-2 because 2 numbers are already printed
  return 0;
 }
```

Output:

```
Enter the number of elements:15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

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

```
/*C program to find LCM of two numbers using recursion */
#include <stdio.h>
/* Function declaration */
int lcm(int a, int b);
int main()
{
    int num1, num2, LCM;
    /* Input two numbers from user */
    printf("Enter any two numbers to find lcm: ");
    scanf("%d%d", &num1, &num2);
    /*Ensures that first parameter of LCM functioni s always less than second */
```

```
    if(num1 > num2)
        LCM = lcm(num2, num1);
    else
        LCM = lcm(num1, num2);
    printf("LCM of %d and %d = %d", num1, num2, LCM);
    return 0;
}

/*Recursive function to find lcm of two numbers 'a' and 'b'.Here 'a' needs to be always less than 'b'.
 */
int lcm(int a, int b)
{
    static int multiple = 0;
    /* Increments multiple by adding max value to it */
    multiple += b;

    /*Base condition of recursion If found a common multiple then return the multiple. */
    if((multiple % a == 0) && (multiple % b == 0))
    {
        return multiple;
    }
    else
    {
        return lcm(a, b);
    }
}
```

Output:

```
Enter any two numbers to find lcm: 12
30
LCM of 12 and 30 = 60
```

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

```
#include<stdio.h>
long int multiplyNumbers(int n);
int main() {
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    printf("Factorial of %d = %ld", n, multiplyNumbers(n));
    return 0;
}

long int multiplyNumbers(int n) {
    if (n>=1)
        return n*multiplyNumbers(n-1);
    else
        return 1;
}
```

Output: Enter a positive integer: 6
Factorial of 6 = 720

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

```
/* C Program to implement Ackermann function using recursion */

#include<stdio.h>
int A(int m, int n);

main()
{
        int m,n;
        printf("Enter two numbers :: \n");
        scanf("%d%d",&m,&n);
        printf("\nOUTPUT :: %d\n",A(m,n));
}

int A(int m, int n)
{
        if(m==0)
                return n+1;
        else if(n==0)
                return A(m-1,1);
  else
return  A(m-1,A(m,n-1));
}
```

Output:

```
Enter two numbers ::
1
0
OUTPUT :: 2
```

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

```
#include <stdio.h>
int addNumbers(int n);
int main() {
  int num;
  printf("Enter a positive integer: ");
  scanf("%d", &num);
  printf("Sum = %d", addNumbers(num));
  return 0;
}
int addNumbers(int n) {
  if (n != 0)
    return n + addNumbers(n - 1);
  else
    return n;
}
```

Output:

```
Enter a positive integer: 20 Sum = 210
```

WEEK 13

Lab 13: Simple functions using Call by reference, Dangling pointers.

i) Write a C program to swap two numbers using call by reference.
ii) Demonstrate Dangling pointer problem using a C program.
iii) Write a C program to copy one string into another using pointer.
iv) Write a C program to find no of lowercase, uppercase, digits and othercharacters using pointers.

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

```
#include <stdio.h>
void swap(int *, int *); //prototype of the function
int main()
{
   int a = 10;
   int b = 20;
   printf("Before swapping the values in main a = %d, b = %d\n",a,b);
   // printing the value of a a nd b in main
   swap(&a,&b);
   printf("After swapping values in main a = %d, b = %d\n",a,b);
   // The values of actual paramet ers do change in call by reference, a = 10, b = 20
}
void swap (int *a, int *b)
{
   int temp;
   temp = *a;
   *a=*b;
   *b=temp;
   printf("After swapping values in function a = %d, b = %d\n",*a,*b);
   // Formal parameters, a = 20, b = 10
}
```

Output

```
Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 20, b = 10
```

ii) Demonstrate Dangling pointer problem using a C program

Dangling Pointers in C

The most common bugs related to pointers and memory management is dangling/wild pointers. Sometimes the programmer fails to initialize the pointer with a valid address, then this type of initialized pointer is known as a dangling pointer in C.

Dangling pointer occurs at the time of the object destruction when the object is deleted or de- allocated from memory without modifying the value of the pointer. In this case, the pointer is pointing to the memory, which is de-allocated. The dangling pointer can point to the memory, which contains either the program code or the code of the operating system

```c
#include <stdio.h>
int main()
{
   int *ptr=(int *)malloc(sizeof(int));
   int a=560;
   ptr=&a;
   free(ptr);
   return 0;
}
```

Output:

Source code:

```c
#include <stdlib.h>

int main()
{
        int x = 4;
        float y = 5.5;

        // A void pointer
        void* ptr;
        ptr = &x;

        // (int*)ptr - does type casting of void
        // *((int*)ptr) dereferences the typecasted
        // void pointer variable.
        printf("Integer variable is = %d", *((int*)ptr));
        // void pointer is now float
        ptr = &y;
        printf("\nFloat variable is = %f", *((float*)ptr));
        return 0;
}
```

Output

Integer variable is = 4

Float variable is = 5.500000

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

```c
#include<stdio.h>
void copy_string(char*, char*);
main()
{
   char source[100], target[100];
   printf("Enter source string\n");
   gets(source);
   copy_string(target, source);
   printf("Target string is \"%s\"\n", target);
   return 0;
}
```

```
void copy_string(char *target, char *source)
{
   while(*source)
   {
      *target = *source;
      source++;
      target++;
   }
   *target = '\0';
}
```

Program Output:

```
Enter source string
w3schools.in
Target string is "w3schools.in"
```

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

```
#include <stdio.h>
int main()
{
   //1
   char inputString[100];
   int upperCount, lowerCount, specialCount, digitCount, i;
   //2
   printf("Enter a String : ");
   gets(inputString);
   //3
   printf("String input is %s ", inputString);
   //4
   upperCount = lowerCount = specialCount = digitCount = 0;
   //5
   for (i = 0; inputString[i] != '\0'; i++)
   {
      //6
      if (inputString[i] >= 'A' && inputString[i] <= 'Z')
      {
         upperCount++;
      }
      else if (inputString[i] >= 'a' && inputString[i] <= 'z')
      {
         lowerCount++;
      }
      else if (inputString[i] >= '0' && inputString[i] <= '9')
      {
         digitCount++;
      }
      else
      {
         specialCount++;}}
```

```
    //7
    printf("\nUpper case count : %d \n", upperCount);
    printf("Lower case count : %d \n", lowerCount);
    printf("Digit count : %d \n", digitCount);
    printf("Special character count : %d \n", specialCount);

    return 0;
}
```

Output: Enter a String : Hello world 112@#$
String input is Hello world 112@#$
Upper case count : 1
Lower case count : 9
Digit count : 3
Special character count : 5

Enter a String : Sample112@#$
String input is Sample112@#$
Upper case count : 1
Lower case count : 5
Digit count : 3
Special character count : 3

WEEK 14

Lab 14: File operations
i) Write a C program to write and read text into a file.
ii) Write a C program to write and read text into a binary file using fread() and fwrite()
iii) Copy the contents of one file to another file.
iv) Write a C program to merge two files into the third file using command-line arguments.
v) Find no. of lines, words and characters in a file
vi) Write a C program to print last n characters of a given file.

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

```
#include< stdio.h >
int main()
{
   FILE *fp;  /* file pointer*/
   char fName[20];
   printf("\nEnter file name to create :");
   scanf("%s",fName);
   /*creating (open) a file*/
   fp=fopen(fName,"w");
   /*check file created or not*/
   if(fp==NULL)
   {
      printf("File does not created!!!");
      exit(0); /*exit  from program*/
   }
   printf("File created successfully.");
   /*writting into file*/
   putc('A',fp);
   putc('B',fp);
   putc('C',fp);
   printf("\nData written successfully.");
   fclose(fp);
   /*again open file to read data*/
   fp=fopen(fName,"r");
   if(fp==NULL)
   {
      printf("\nCan't open file!!!");
      exit(0);
   }
   printf("Contents of file is :\n");
   printf("%c",getc(fp));
   printf("%c",getc(fp));
   printf("%c",getc(fp));
   fclose(fp);
   return 0;
}
```

Output:

```
Enter file name to create : ok.txt
File created successfully.
Data written successfully.
Contents of file is :
```

ABC

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

```
#include<stdio.h>
void main()
{
   FILE *fp = NULL;
   short x[10] = {1,2,3,4,5,6,5000,6,-10,11};
   short result[10];

   fp=fopen("c:\\temp.bin", "wb");

   if(fp != NULL)
   {
      fwrite(x, 2 /*sizeof(short)*/, 10 /*20/2*/, fp);
      rewind(fp);
      fread(result, 2 /*sizeof(short)*/, 10 /*20/2*/, fp);
   }
   else
      exit(0);

   printf("\nResult");
   printf("\n%d",result[0]);
   printf("\n%d",result[1]);
   printf("\n%d",result[2]);
   printf("\n%d",result[3]);
   printf("\n%d",result[4]);
   printf("\n%d",result[5]);
   printf("\n%d",result[6]);
   printf("\n%d",result[7]);
   printf("\n%d",result[8]);
   printf("\n%d",result[9]);

   fclose(fp)
}
```

Output:

After I do the fread() (HEX values): temp.bin:
01 02 03 04 05 06 e1 8e 88 06 ef bf b6 0b...
After I do the fwrite()
stdout:
Resul
t 0
914
-28
-28714
-32557
1
512
-32557
908
914

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

```
#include<stdio.h>
int main()
{
    int ch;
    FILE *fp,*fq;
    fp=fopen("source.txt","r");
    fq=fopen("backup.txt","w");
    if(fp==NULL||fq==NULL)
        printf("File does not exist..");
    else
        while((ch=fgetc(fp))!=EOF)
        {
            fputc(ch,fq);
        }
    printf("File copied.....");
    return 0;
}
```

Output:

```
File copied.....
```

source.txt

```
Welcome to TutorialRide..

Happy c-programming

Created By..

    Sapna
```

source.txt | backup.txt

```
Welcome to TutorialRide..

Happy c-programming

Created By..

    Sapna
```

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 *fs1, *fs2, *ft;
	char ch, file1[20], file2[20], file3[20];


	if ( argc != 4 )
	{
		printf("There is no file names..\n");
		exit(0);
	}
	fs1 = fopen(argv[1],"r");
	fs2 = fopen(argv[2],"r");
	if( fs1 == NULL || fs2 == NULL )
	{
		perror("Error ");
		printf("Press any key to exit...\n");
		exit(0);
		exit(EXIT_FAILURE);
	}
	ft = fopen(argv[3],"w");
	if( ft == NULL )
	{
		perror("Error ");
		printf("Press any key to exit...\n");
		exit(EXIT_FAILURE);
	}
	while( ( ch = fgetc(fs1) ) != EOF )
		fputc(ch,ft);
	while( ( ch = fgetc(fs2) ) != EOF )
		fputc(ch,ft);
	printf("Two files were merged into %s file successfully.\n",argv[3]);

	fclose(fs1);
	fclose(fs2);
	fclose(ft);

	return 0;
}
```

**********OUTPUT**********
********** christo@christo-System-Product-Name:~/Documents/programs$ gcc file.c christo@christo-System-Product-Name:~/Documents/programs$ ./a.out 1.txt 2.txt 3.txt Two files were merged into 3.txt file successfully.

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

```
//C code
#include <stdio.h>
int main()
{
    char str[100];//input string with size 100
    int words=0,newline=0,characters=0; // counter variables
    scanf("%[^~]",&str);//scanf formatting
    for(int i=0;str[i]!='\0';i++)
     {
        if(str[i] == ' ')
        {
            words++;
        }
        else if(str[i] == '\n')
        {
           newline++;
            words++;//since with every next line new words start. corner case 1
        }
        else if(str[i] != ' ' && str[i] != '\n'){
        characters++;
        }
     }
    if(characters > 0)//Corner case 2,3.
    {
       words++;
       newline++;
    }
     printf("Total number of words : %d\n",words);
    printf("Total number of lines : %d\n",newline);
    printf("Total number of characters : %d\n",characters);
    return 0;
}
```

Output:

Deepak Kumar is the brother of Aditya Kumar~

Output : Total number of words : 8
Total number of lines : 2
Total number of characters : 36

vi) Write a C program to print last n characters of a given file.

```
#include<stdio.h>
voidmain()
{
FILE*fp;
char ch;
int n;
long len;
clrscr();
printf("Enter the value of n : ");
scanf("%d",&n);
fp=fopen("test.txt","r");
if(fp==NULL)
```

```
{
puts("cannot open this file");
exit(1);
}
fseek(fp,0,SEEK_END);
len =ftell(fp);
fseek(fp,(len-n),SEEK_SET);
do{
  ch =fgetc(fp);
putchar(ch);
}while(ch!=EOF);
fclose(fp);
getch();
}
```

Output :

```
Enter the value of n : 4
.com
```