GOVERNMENT POLYTECHNIC SHEOHAR

LABORATORY MANUAL

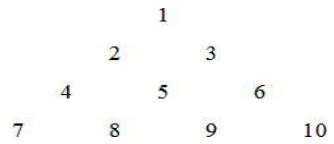
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

II SEMESTER

PROGRAMMING WITH 'C'

SUBJECT CODE: P2418101

LIST OF EXPERIMENTS

1.	(a) Develop minimum 3 programs using Constants, Variables, arithmetic expression. (b) Develop minimum 3 programs to exhibiting use of increment/decrement operators, data type conversion
2.	(a) Write simple program to convert temperature in Fahrenheit degrees to Centigrade degrees. (take input from the user) (b) Write simple programs to calculate the area and perimeter of the rectangle, and the area & circumference of the circle (take input from the user)
3.	Write program to: (a) Determine whether a given year is a leap year or not. (b) Determine whether a string is palindrome. (c) Find the greatest of the three numbers using conditional operators (d) Find if a given character is vowel (use if-else ladder).
4.	Using switch statement- Write program to: (a) Print day of week by taking number from 1 to 7. (b) Print a student's grade ("A", "B", "C" etc.) by accepting his/her percent marks. (c) Find if a given character is vowel. (d) Using "if" and "switch" statements- Write programs to check whether the triangle is equilateral, isosceles, or scalene triangle.
5.	Write Program to: (a) Find sum of digits of a given number. (b) Generate multiplication table up to 10 for numbers 1 to 5. (c) Find Fibonacci series for given number. (d) Write a program to produce the following output: 
6	Develop a Program to: (a) Sort list of 10 numbers. (b) Perform addition of 3x3 matrix.

7	Develop Program to: (a) Create a structure called "library" to hold details of a book viz. accession number, title of the book, author name, price of the book, and flag indicating whether book is issued or not. Fetch some sample data and display the same. (b) Develop and execute C Program to Add Two Distances given in kilometer- meter Using Structures.
8	Develop Program to demonstrate: (a) Use of all String handling functions. (b) Use of few Mathematical functions. (c) Use of few other miscellaneous functions. Develop a Program to: (a) Create a function to find GCD of given number. Call this function in a program. (b) Find Factorial of given number using recursion.
9	(a) Develop a Program to Print values of variables and their addresses. (b) Develop a Program to Find sum of all elements stored in given array using pointers.

Course Outcomes (COs):-

CO-1 : Develop flowchart and algorithm to solve problems logically.

CO-2 : Write simple 'C' programs using arithmetic expressions.

CO-3 : Develop 'C' programs using control structure.

CO-4 : Develop 'C' programs using arrays and structures.

CO-5 : Develop/Use functions in C programs for modular programming approach.

CO-6 : Develop 'C' programs using pointers.

Program Outcomes (POs):-

PO1 : Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

PO2 : Problem analysis: Identify and analyse well-defined engineering problems using codified standard methods.

PO3 : Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4 : Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

PO5 : Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.

PO6 : Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well- defined engineering activities.

PO7 : Life-long learning: Ability to analyze individual needs and engage in updating in the context of technological changes.

Program Specific Outcomes (PSOs):-

PSO1: Apply the fundamentals of computer science, including algorithms and programming, to analyze complex problems and develop robust solutions.

PSO2: Design and implement computer science solutions with emphasis on energy efficiency, sustainability, and eco-friendly technologies.

MAPPING

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PSO1	PSO2	
CO1	1	2	1	-	-	-	-	3	1	**AP**
CO2	1	1	1	1	-	-	-	2	1	**AP**
CO3	1	2	1	1	-	-	-	2	1	**AP**
CO4	1	2	1	1	-	-	-	2	1	**AP**
CO5	1	2	2	1	-	-	-	3	1	**AP**
CO6	1	2	3	1	-	-	-	3	1	**AP**

EXPERIMENT NO: 1

1.1 Develop minimum 3 programs using Constants, Variables, arithmetic expression.

1.1.1 Program to Calculate the Area of a Rectangle.

AIM: - To write a C program to calculate the area of a rectangle using constants, variables, and arithmetic expressions.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Area of a rectangle is calculated using the arithmetic expression:

Area=Length×Breadth

Constants store fixed values, variables store input, and the arithmetic operator * is used for multiplication.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare variables for length and breadth.

Step 3: Read input values.

Step 4: Use the arithmetic expression to compute the area.

Step 5: Display the result.

Step 6: Stop the program.

PROGRAM:-

```
#include <stdio.h>
#define UNIT "square units"   // Constant
int main() {
    float length, breadth, area;
    printf("Enter length of rectangle: ");
    scanf("%f", &length);
    printf("Enter breadth of rectangle: ");
```

```
    scanf("%f", &breadth);
    area = length * breadth;  // Arithmetic expression
    printf("Area of Rectangle = %.2f %s", area, UNIT);
    return 0;
}
```

OUTPUT:-

```
Enter length of rectangle: 5
Enter breadth of rectangle: 3
Area of Rectangle = 15.00 square units

=== Code Execution Successful ===
```

RESULT:-

The program successfully calculates the area of a rectangle.

1.1.2 Program to Convert Temperature from Celsius to Fahrenheit.

AIM: - To write a C program to convert temperature from Celsius to Fahrenheit using variables and constants.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Temperature conversion formula:

$F=(C\times1.8)+32$

Here, 1.8 and 32 are constants. The arithmetic operators * and + are used.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare variables for Celsius and Fahrenheit.

Step 3: Take input from the user.

Step 4: Apply formula using arithmetic expression.

Step 5: Display result.

Step 6: Stop program.

PROGRAM:-

```
#include <stdio.h>
#define FACTOR 1.8   // Constant
#define OFFSET 32    // Constant
int main() {
    float celsius, fahrenheit;
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);
    fahrenheit = (celsius * FACTOR) + OFFSET;  // Arithmetic expression
    printf("Temperature in Fahrenheit = %.2f", fahrenheit);
    return 0;

}
```

OUTPUT:-

```
Enter temperature in Celsius: 25
Temperature in Fahrenheit = 77.00

=== Code Execution Successful ===
```

RESULT:-

The program successfully converts Celsius temperature to Fahrenheit.

1.1.3 Program to Calculate Simple Interest.

AIM: - To write a C program to calculate simple interest using constants, variables, and arithmetic expressions.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Formula for Simple Interest:

SI=P×T×R/100

Where:

- P = Principal amount
- T = Time
- R = Rate of interest

  Arithmetic operators: * for multiplication and / for division.

PROCEDURE:-

Step 1: Declare variables for P, T, R, and SI.

Step 2: Input values from user.

Step 3: Apply arithmetic formula.

Step 4: Display the result.

Step 5: End program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    float principal, time, rate, si;
    printf("Enter Principal Amount: ");
    scanf("%f", &principal);
    printf("Enter Time (in years): ");
    scanf("%f", &time);
    printf("Enter Rate of Interest: ");
    scanf("%f", &rate);
```

```
    si = (principal * time * rate) / 100;  // Arithmetic expression
    printf("Simple Interest = %.2f", si);
    return 0;
}
```

OUTPUT:-

```
Enter Principal Amount: 1000
Enter Time (in years): 2
Enter Rate of Interest: 5
Simple Interest = 100.00

=== Code Execution Successful ===
```

RESULT:-

The program successfully calculates the simple interest.

1.2 Develop minimum 3 programs to exhibiting use of increment/decrement operators, data type conversion.

1.2.1 Use of Increment Operator (++)

AIM: - To write a C program that demonstrates the use of the increment operator (++).

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Increment operator (++) increases a variable's value by **1**.

Two types:

- **Pre-increment:** ++x (increments first, then uses the value)
- **Post-increment:** x++ (uses the value, then increments)

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare integer variables.

Step 3: Apply pre-increment and post-increment operators.

Step 4: Display results.

Step 5: Compile and run the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    int a = 10, b;
    b = ++a; // pre-increment
    printf("After pre-increment (++a): a = %d, b = %d\n", a, b);
    b = a++; // post-increment
    printf("After post-increment (a++): a = %d, b = %d\n", a, b);
    return 0;
}
```

OUTPUT:-

```
After pre-increment (++a): a = 11, b = 11
After post-increment (a++): a = 12, b = 11


=== Code Execution Successful ===
```

RESULT:-

The program successfully demonstrated pre-increment and post-increment operators.

1.2.2 Use of Decrement Operator (--)

AIM: - To develop a C program that demonstrates the decrement operator (--).

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Decrement operator (--) decreases a variable's value by **1**.

Types:

- Pre-decrement: --x
- Post-decrement: x--

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare integer variables.

Step 3: Apply pre-decrement and post-decrement.

Step 4: Display results.

Step 5: Compile and run the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    int x = 20, y;
    y = --x;  // pre-decrement
    printf("After pre-decrement (--x): x = %d, y = %d\n", x, y);
    y = x--;  // post-decrement
    printf("After post-decrement (x--): x = %d, y = %d\n", x, y);
    return 0;
}
```

OUTPUT:-

```
After pre-decrement (--x): x = 19, y = 19
After post-decrement (x--): x = 18, y = 19


=== Code Execution Successful ===
```

RESULT:-

The program successfully exhibited pre-decrement and post-decrement operations.

1.2.3 Data Type Conversion (Casting)

AIM: - To write a C program showing data type conversion (implicit and explicit).

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Data type conversion allows changing one data type into another.

Two types:

- Implicit conversion: done automatically by compiler
- Explicit conversion (Casting): done manually using (type)

Example:

float x = (float) a / b;

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare integer variables.

Step 3: Perform division without casting (integer division).

Step 4: Perform division with casting (floating-point division).

Step 5: Display results.

Step 6: Compile and run the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
   int a = 5, b = 2;
   float result1, result2;
   result1 = a / b;          // integer division
   result2 = (float)a / b;     // explicit type casting
   printf("Without casting (a/b): %f\n", result1);
   printf("With casting ((float)a/b): %f\n", result2);
   return 0; }
```

OUTPUT:-

```
Without casting (a/b): 2.000000
With casting ((float)a/b): 2.500000


=== Code Execution Successful ===
```

RESULT:-

The program successfully demonstrated implicit and explicit type conversion in C.

EXPERIMENT NO: 2

2.1 Write simple program to convert temperature in Fahrenheit degrees to Centigrade degrees. (take input from the user).

AIM: - To write a simple C program to convert temperature given in Fahrenheit to Centigrade by taking input from the user.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Temperature can be measured in different scales.

The most commonly used are:

- Fahrenheit (°F)
- Centigrade / Celsius (°C)

To convert temperature from Fahrenheit to Centigrade, we use the formula:

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

Where,

- F = Temperature in Fahrenheit
- C = Temperature in Centigrade

The program will:

1. Take Fahrenheit value from the user.
2. Apply the conversion formula.
3. Display the temperature in Centigrade.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare variables for Fahrenheit and Centigrade.

Step 3: Ask the user to enter temperature in Fahrenheit.

Step 4: Read user input.

Step 5: Apply the conversion formula:

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

Step 6: Display the result.

Step 7: End the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    float fahrenheit, centigrade;
    printf("Enter temperature in Fahrenheit: ");
    scanf("%f", &fahrenheit);
    centigrade = (5.0 / 9.0) * (fahrenheit - 32);
    printf("Temperature in Centigrade = %.2f\n", centigrade);
    return 0; }
```

OUTPUT:-

```
Enter temperature in Fahrenheit: 99
Temperature in Centigrade = 37.22


=== Code Execution Successful ===
```

RESULT:-

The program successfully reads the temperature in Fahrenheit from the user, converts it into Centigrade, and displays the converted temperature.

2.2 Write simple programs to calculate the area and perimeter of the rectangle, and the area & circumference of the circle (take input from the user).

AIM: - To write C programs that take input from the user and calculate:

The area and perimeter of a rectangle.

The area and circumference of a circle.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Rectangle

For a rectangle:

- Area = length × breadth
- Perimeter = 2 × (length + breadth)

Rectangle is a 2D shape with opposite sides equal and four right angles.

Circle

For a circle:

- Area = $\pi \times r^2$
- Circumference = $2 \times \pi \times r$

Where:

- r = radius
- π = 3.14 (approximately)

PROCEDURE:-

Step 1: Start the program.

Step 2: Include the necessary header files (stdio.h).

Step 3: Declare variables for length, breadth, radius, area, perimeter, and circumference.

Step 4: Take user input for

- length and breadth (for rectangle)
- radius (for circle)

Step 5: Apply formulas to calculate the required values.

Step 6: Display the calculated values to the user.

Step 7: End the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    float length, breadth, radius;
    float rectangleArea, rectanglePerimeter;
    float circleArea, circleCircumference;
    float pi = 3.14;
    /* Rectangle Input */
    printf("Enter length of the rectangle: ");
    scanf("%f", &length);
    printf("Enter breadth of the rectangle: ");
    scanf("%f", &breadth);
    /* Rectangle Calculations */
    rectangleArea = length * breadth;
    rectanglePerimeter = 2 * (length + breadth);
    /* Circle Input */
    printf("\nEnter radius of the circle: ");
    scanf("%f", &radius);
    /* Circle Calculations */
    circleArea = pi * radius * radius;
    circleCircumference = 2 * pi * radius;
    printf("\n----- RESULTS ------\n");
    printf("Rectangle Area: %.2f\n", rectangleArea);
    printf("Rectangle Perimeter: %.2f\n", rectanglePerimeter);
    printf("Circle Area: %.2f\n", circleArea);
    printf("Circle Circumference: %.2f\n", circleCircumference);
    return 0;
}
```

OUTPUT:-

```
Enter length of the rectangle: 10
Enter breadth of the rectangle: 5

Enter radius of the circle: 7

----- RESULTS -----
Rectangle Area: 50.00
Rectangle Perimeter: 30.00
Circle Area: 153.86
Circle Circumference: 43.96


=== Code Execution Successful ===
```

RESULT:-

The C program successfully accepts user input and calculates:

- Area and Perimeter of the Rectangle
- Area and Circumference of the Circle

The output matches the expected mathematical results.

EXPERIMENT NO: 3

3.1 Write program to Determine whether a given year is a leap year or not.

AIM: - To write a C program to determine whether a given year is a leap year or not.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

A leap year is a year that has 366 days instead of 365.

The extra day is added to the month of February (29 days).

A year is considered a leap year if it fulfills the following conditions:

1. The year is divisible by 4
2. But if the year is also divisible by 100, then it must also be divisible by 400 to be a leap year.

Leap Year Rules

- If year % 400 == 0 → Leap Year
- Else if year % 100 == 0 → Not a Leap Year
- Else if year % 4 == 0 → Leap Year
- Else Not a Leap Year

Example:

- 2020 → Leap year (divisible by 4)
- 1900 → Not a leap year (divisible by 100 but not 400)
- 2000 → Leap year (divisible by 400)

PROCEDURE:-

Step 1: Start the program.

Step 2: Include necessary header files.

Step 3: Declare a variable to store the year.

Step 4: Accept the year from the user.

Step 5: Apply leap year conditions using if-else statements.

Step 6: Display whether the year is a leap year or not.

Step 7: End the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    int year;
    printf("Enter a year: ");
    scanf("%d", &year);
    if (year % 400 == 0) {
        printf("%d is a Leap Year.\n", year); }
    else if (year % 100 == 0) {
        printf("%d is Not a Leap Year.\n", year); }
    else if (year % 4 == 0) {
        printf("%d is a Leap Year.\n", year); }
    else {
        printf("%d is Not a Leap Year.\n", year); }
    return 0; }
```

OUTPUT:-

```
Enter a year: 2024
2024 is a Leap Year.


=== Code Execution Successful ===
```

```
Enter a year: 1990
1990 is Not a Leap Year.


=== Code Execution Successful ===
```

RESULT:-

The program successfully determines whether a given year is a leap year or not based on standard leap year rules.

3.2 Write program to Determine whether a string is palindrome.

AIM: - To write a C program to check whether a given string is a palindrome.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

A palindrome is a word, number, or string that reads the same forward and backward.

Examples of palindrome strings:

- "madam"
- "racecar"
- "level"

To check if a string is palindrome:

1. Compare the first and last characters.
2. Move inward (second with second-last…).
3. If all matched → string is palindrome.
4. If any character mismatches → not a palindrome.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare a string variable.

Step 3: Input a string from the user.

Step 4: Find the length of the string.

Step 5: Use a loop to compare characters from the beginning and end.

Step 6: If all characters match → string is palindrome.

Step 7: Display the result.

Step 8: Stop the program.

PROGRAM:-

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[100];
```

```
int i, len, flag = 0;
printf("Enter a string: ");
fgets(str, sizeof(str), stdin);
str[strcspn(str, "\n")] = '\0';
len = strlen(str);
for (i = 0; i < len / 2; i++) {
    if (str[i] != str[len - i - 1]) {
        flag = 1;
        break; } }
if (flag == 0) {
    printf("The string \"%s\" is a palindrome.\n", str);
} else {
    printf("The string \"%s\" is not a palindrome.\n", str); }
return 0;  }
```

OUTPUT:-

```
Enter a string: sir
The string "sir" is not a palindrome.


=== Code Execution Successful ===
```

```
Enter a string: maam
The string "maam" is a palindrome.


=== Code Execution Successful ===
```

RESULT:-

The C program was successfully executed to determine whether a given string is a palindrome. The program correctly checks and displays whether the input string reads the same forward and backward.

3.3 Write program to Find the greatest of the three numbers using conditional operators.

AIM: - To write a C program to check whether a given string is a palindrome.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

The conditional operator (?:) is a compact alternative to an if-else statement.

Its syntax is:

condition ? expression_if_true : expression_if_false;

For comparing three numbers, nested conditional operators can be used to check the greatest number among them.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare three integer variables to store numbers.

Step 3: Input the three numbers from the user.

Step 4: Use nested conditional operators to find the greatest number.

Step 5: Display the largest number on the screen.

Step 6: End the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    int a, b, c, greatest;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);
    greatest = (a > b) ?
                (a > c ? a : c) :
                (b > c ? b : c);
    printf("The greatest number is: %d\n", greatest);
```

return 0; }

OUTPUT:-

```
Enter three numbers: 12 45 28
The greatest number is: 45


=== Code Execution Successful ===
```

RESULT:-

The C program successfully finds and displays the greatest of the three numbers using the conditional (ternary) operator.

3.4 Write program to Find if a given character is vowel (use if-else ladder).

AIM: - To write a C program to check whether a given character is a vowel or consonant using an if-else ladder.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

A character in English alphabets is considered a vowel if it is any one of the following letters: a, e, i, o, u (or their uppercase forms A, E, I, O, U).

All other alphabetic characters are consonants.

Using an if-else ladder, we compare the input character with each vowel.

- If it matches, it is a vowel
- Otherwise, it is a consonant

The program uses conditional statements to perform character comparison.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare a character variable.

Step 3: Accept a character from the user.

Step 4: Use an if-else ladder to compare the character with vowel letters.

Step 5: If the character matches any vowel, display “Vowel”.

Step 6: Else, display “Consonant”.

Step 7: End the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    char ch;
    printf("Enter a character: ");
    scanf("%c", &ch);
    if (ch == 'a' || ch == 'A')
```

```
    printf("%c is a vowel.\n", ch);
else if (ch == 'e' || ch == 'E')
    printf("%c is a vowel.\n", ch);
else if (ch == 'i' || ch == 'I')
    printf("%c is a vowel.\n", ch);
else if (ch == 'o' || ch == 'O')
    printf("%c is a vowel.\n", ch);
else if (ch == 'u' || ch == 'U')
    printf("%c is a vowel.\n", ch);
else
    printf("%c is a consonant.\n", ch);
return 0; }
```

OUTPUT:-

```
Enter a character: e
e is a vowel.


=== Code Execution Successful ===
```

```
Enter a character: s
s is a consonant.


=== Code Execution Successful ===
```

RESULT:-

The program successfully checks whether the entered character is a vowel or consonant using an if-else ladder.

EXPERIMENT NO: 4

4.1 Using switch statement - Write program to:

4.1.1 Print day of week by taking number from 1 to 7

AIM: - To write a C program using a switch statement to print the day of the week by taking a number from 1 to 7 as input.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In C programming, the switch statement is a multi-way decision-making statement. It allows the execution of different blocks of code based on the value of an expression.

In this program:

- The user enters a number from 1 to 7.
- Each number corresponds to a day of the week.
- The switch statement compares the input value with different cases and prints the appropriate day.
- The default case is used to handle invalid inputs.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare an integer variable.

Step 3: Display a message asking the user to enter a number between 1 and 7.

Step 4: Read the input number using scanf().

Step 5: Use a switch statement to check the input value.

Step 6: Print the corresponding day of the week.

Step 7: If the number is not between 1 and 7, display an error message.

Step 8: Stop the program.

PROGRAM:-

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

```
{
    int day;
    printf("Enter a number (1 to 7): ");
    scanf("%d", &day);
    switch(day)
    {
        case 1:
            printf("Sunday");
            break;
        case 2:
            printf("Monday");
            break;
        case 3:
            printf("Tuesday");
            break;
        case 4:
            printf("Wednesday");
            break;
        case 5:
            printf("Thursday");
            break;
        case 6:
            printf("Friday");
            break;
        case 7:
            printf("Saturday");
            break;
        default:
            printf("Invalid input! Please enter a number between 1 and 7.");
    }
    return 0;
}
```

OUTPUT:-

```
Enter a number (1 to 7): 5
Thursday

=== Code Execution Successful ===
```

RESULT:-

The program successfully prints the day of the week corresponding to the number entered by the user using a switch statement.

4.1.2 Print a student's grade ("A", "B", "C" etc.) by accepting his/her percent marks.

AIM: - To write a c program that accepts percentage marks of a student and prints the corresponding grade using the switch statement.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In C programming, the switch statement is a multi-way decision-making control structure. It allows execution of different blocks of code based on the value of an expression.

In this program:

- The student's percentage marks are divided by 10 to obtain a simplified value.
- The result is used in a switch case to decide the grade.

Grade Criteria:

Percentage	Grade
90 – 100	A
80 – 89	B
70 – 79	C
60 – 69	D
Below 60	F

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare required variables.

Step 3: Accept percentage marks from the user.

Step 4: Divide percentage by 10.

Step 5: Use switch statement to match grade cases.

Step 6: Display the corresponding grade.

Step 7: Stop the program.

PROGRAM:-

```
#include <stdio.h>
int main()
{  int percent, grade;
   printf("Enter the percentage marks: ");
   scanf("%d", &percent);
   grade = percent / 10;
   switch (grade) {
      case 10:
      case 9:
         printf("Grade: A");
         break;
      case 8:
         printf("Grade: B");
         break;
      case 7:
         printf("Grade: C");
         break;
      case 6:
         printf("Grade: D");
         break;
      default:
         printf("Grade: F"); }
   return 0; }
```

OUTPUT:-

```
Enter the percentage marks: 90
Grade: A

=== Code Execution Successful ===
```

RESULT:-

The program successfully accepts the student's percentage marks and displays the correct grade using the switch statement.

4.1.3 Print a Find if a given character is vowel.

AIM :- To write a C program using a switch statement to find whether a given character is a vowel or a consonant.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In the C programming language, the switch statement is a decision-making control structure used to execute different blocks of code based on the value of an expression.

Vowels in the English alphabet are:

- Lowercase: a, e, i, o, u
- Uppercase: A, E, I, O, U

Using a switch statement, the input character is compared with each vowel case.

If a match is found, the character is identified as a vowel; otherwise, it is considered a consonant.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare a character variable.

Step 3: Accept a character from the user.

Step 4: Use a switch statement to compare the character with vowel cases.

Step 5: Print whether the character is a vowel or a consonant.

Step 6: Stop the program.

PROGRAM:-

```
#include <stdio.h>
int main()
{
   char ch;
   printf("Enter a character: ");
   scanf("%c", &ch);
   switch (ch)
   {
```

```
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U':
            printf("The given character is a Vowel.");
            break;
        default:
            printf("The given character is a Consonant.");
    }
    return 0;
}
```

OUTPUT:-

```
Enter a character: a
The given character is a Vowel.

=== Code Execution Successful ===
```

```
Enter a character: t
The given character is a Consonant.

=== Code Execution Successful ===
```

RESULT:-

Thus, the C program using a switch statement to check whether a given character is a vowel or a consonant was written, compiled, and executed successfully.

4.1.4 Write programs to check whether the triangle is equilateral, isosceles, or scalene triangle Using "if" and "switch" statements

AIM :- To write and execute C programs to check whether a given triangle is Equilateral, Isosceles, or Scalene using

1. If - Else statement
2. switch statement

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

A triangle can be classified based on the lengths of its three sides:

- Equilateral Triangle: All three sides are equal
- Isosceles Triangle: Any two sides are equal
- Scalene Triangle: All three sides are different

In C programming:

- if–else statements are used for conditional decision-making.
- switch statements are used when multiple conditions depend on a single expression.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare three integer variables to store the sides of the triangle.

Step 3: Read values of the three sides from the user.

Step 4: Compare the sides using:

- if–else conditions (Program 1)
- switch statement (Program 2)

Step 5: Display the type of triangle.

Step 6: Stop the program.

PROGRAM 1 :-

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

```
    int a, b, c;
    printf("Enter three sides of the triangle: ");
    scanf("%d %d %d", &a, &b, &c);
    if (a == b && b == c)
    {
        printf("Triangle is Equilateral");
    }
    else if (a == b || b == c || a == c)
    {
        printf("Triangle is Isosceles");
    }
    else
    {
        printf("Triangle is Scalene");
    }
    return 0;
}
```

OUTPUT 1:-

```
Enter three sides of the triangle: 2
5
9
Triangle is Scalene

=== Code Execution Successful ===
```

PROGRAM 2:-

```
 #include <stdio.h>
int main()
{
    int a, b, c, choice;
    printf("Enter three sides of the triangle: ");
    scanf("%d %d %d", &a, &b, &c);
```

```
if (a == b && b == c)
    choice = 1;
else if (a == b || b == c || a == c)
    choice = 2;
else
    choice = 3;
switch (choice)
{
    case 1:
        printf("Triangle is Equilateral");
        break;
    case 2:
        printf("Triangle is Isosceles");
        break;
    case 3:
        printf("Triangle is Scalene");
        break;
    default:
        printf("Invalid input");
}
return 0; }
```

OUTPUT 2:-

```
Enter three sides of the triangle: 6
6
2
Triangle is Isosceles

=== Code Execution Successful ===
```

RESULT:-

Thus, the C programs to determine whether a triangle is Equilateral, Isosceles, or Scalene using if–else and switch statements were successfully written, compiled, and executed.

EXPERIMENT NO: 5

5.1 Write Program to Find sum of digits of a given number.

AIM: - To write and execute a C program to find the sum of digits of a given number.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In C programming, numbers can be processed digit by digit using arithmetic operations. To find the sum of digits of a number:

- The modulus operator (%) is used to extract the last digit of the number.
- The division operator (/) removes the last digit.
- This process is repeated using a loop until the number becomes zero.

This technique is widely used in numerical computations and forms the basis for many number-based algorithms.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare required variables.

Step 3: Read an integer number from the user.

Step 4: Initialize the sum variable to zero.

Step 5: Use a while loop to:

- Extract the last digit using % 10.
- Add the digit to the sum.
- Remove the last digit using / 10.

Step 6: Display the sum of digits.

Step 7: Stop the program.

PROGRAM:-

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

```
    int number, sum = 0, digit;
    printf("Enter a number: ");
    scanf("%d", &number);
    while (number != 0)
    {
        digit = number % 10;
        sum = sum + digit;
        number = number / 10;
    }
    printf("Sum of digits = %d", sum);
    return 0;
}
```

OUTPUT:-

```
Enter a number: 123
Sum of digits = 6

=== Code Execution Successful ===
```

RESULT:-

Thus, the C program to find the sum of digits of a given number was executed successfully and the correct output was obtained.

5.2 Write Program to Generate multiplication table up to 10 for numbers 1 to 5.

AIM :- To write and execute a C program that generates multiplication tables from 1 to 5, each table printed up to 10 multiples.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

A multiplication table displays the products of a number with a sequence of integers. In this program:

- The outer loop controls the numbers from 1 to 5.
- The inner loop controls the multiplier from 1 to 10.
- Each iteration prints the product of the two loop variables.

The use of nested for loops helps generate multiple tables efficiently.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare integer variables.

Step 3: Use an outer for loop for numbers from 1 to 5.

Step 4: Use an inner for loop for multiplication from 1 to 10.

Step 5: Print the multiplication result using printf().

Step 6: Repeat the process for all numbers.

Step 7: Stop the program.

PROGRAM:-

```
 #include <stdio.h>
int main()
{
   int i, j;
   for (i = 1; i <= 5; i++)
   {
      printf("\nMultiplication Table of %d\n", i);
```

```
        for (j = 1; j <= 10; j++)
        {
            printf("%d x %d = %d\n", i, j, i * j);
        }
    }
    return 0;
}
```

OUTPUT:-

```
Multiplication Table of 1
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Multiplication Table of 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

Multiplication Table of 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
```

```
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Multiplication Table of 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

Multiplication Table of 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

RESULT:-

The C program was successfully executed and displayed multiplication tables from 1 to 5, each up to 10 multiples, using nested loops.

5.3 Write Program to Find Fibonacci series for given number.

AIM :- To write and execute a C program to generate the Fibonacci series for a given number of terms.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

The Fibonacci series is a sequence of numbers in which each number is the sum of the two preceding numbers.

The series starts with:

0, 1

The general form of the Fibonacci series is:

0, 1, 1, 2, 3, 5, 8, 13, ...

Formula:

$F(n) = F(n-1) + F(n-2)$

Where:

- $F(0) = 0$
- $F(1) = 1$

This concept is widely used in mathematics, algorithms, and IoT applications such as sensor data prediction and signal processing.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare integer variables.

Step 3: Read the number of terms from the user.

Step 4: Initialize the first two Fibonacci numbers as 0 and 1.

Step 5: Display the first two numbers.

Step 6: Use a for loop to calculate the remaining terms.

Step 7: Print each term of the series.

Step 8: Stop the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    int n, i;
    int first = 0, second = 1, next;
    printf("Enter the number of terms: ");
    scanf("%d", &n);
    printf("Fibonacci Series: ");
    for (i = 1; i <= n; i++) {
        if (i == 1) {
            printf("%d ", first);
            continue;  }
        if (i == 2) {
            printf("%d ", second);
            continue;  }
        next = first + second;
        first = second;
        second = next;
        printf("%d ", next); }
    return 0; }
```

OUTPUT:-

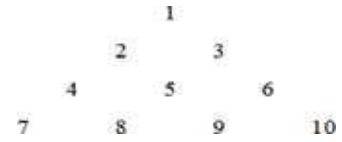
```
Enter the number of terms: 7
Fibonacci Series: 0 1 1 2 3 5 8

=== Code Execution Successful ===
```

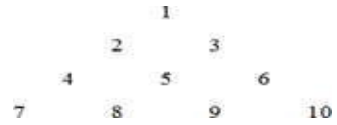
RESULT:-

The program was successfully executed and the Fibonacci series for the given number of terms was generated correctly using the C programming language.

5.4 Write a program to produce the following output



AIM :- To write and execute a C program to print the following number pattern:



SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

The given pattern is known as Floyd's Triangle.

It is a right-angled triangular arrangement of natural numbers, where:

- Numbers are printed continuously.
- Each row contains one more number than the previous row.
- Nested loops are used:
  - The outer loop controls the number of rows.
  - The inner loop prints numbers in each row.

This pattern is useful for understanding loop control, counters, and formatted output in C programming.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare integer variables for rows and number counter.

Step 3: Initialize the counter to 1.

Step 4: Read the number of rows from the user.

Step 5: Use an outer for loop for rows.

Step 6: Use an inner for loop to print numbers in each row.

Step 7: Increment the counter after printing each number.

Step 8: Move to the next line after each row.

Step 9: Stop the program.

PROGRAM:-

```
#include <stdio.h>
int main() {
    int rows, i, j;
    int num = 1;
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    for (i = 1; i <= rows; i++) {
        for (j = 1; j <= i; j++) {
            printf("%d ", num);
            num++;
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT:-

```
Enter number of rows: 4
1
2 3
4 5 6
7 8 9 10


=== Code Execution Successful ===
```

RESULT:-

The program was successfully executed and the required number pattern (Floyd's Triangle) was printed correctly using the C programming language.

EXPERIMENT NO: 6

6.1 Develop a Program to Sort list of 10 numbers.

AIM: - To develop a C program to sort a list of 10 numbers in ascending order using the Bubble Sort technique.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Sorting is the process of arranging data in a specific order such as ascending or descending. Bubble Sort is a simple sorting algorithm that repeatedly compares adjacent elements and swaps them if they are in the wrong order. This process is repeated until the list is completely sorted.

Advantages of Bubble Sort:

- Easy to understand and implement
- Suitable for small datasets

Disadvantage:

- Less efficient for large data sets

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare an integer array of size 10.

Step 3: Read 10 numbers from the user.

Step 4: Apply Bubble Sort logic:

- Compare adjacent elements.
- Swap if the first number is greater than the next.

Step 5: Repeat the process until the array is sorted.

Step 6: Display the sorted list.

Step 7: Stop the program.

PROGRAM:-

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

```
    int a[10], i, j, temp;
    printf("Enter 10 numbers:\n");
    for (i = 0; i < 10; i++) {
        scanf("%d", &a[i]);
    }
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 9 - i; j++) {
            if (a[j] > a[j + 1]) {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
    printf("Sorted list in ascending order:\n");
    for (i = 0; i < 10; i++) {
        printf("%d ", a[i]);
    }
    return 0;
}
```

OUTPUT:-

```
Enter 10 numbers:
45 12 89 23 10 67 34 1 90 5
Sorted list in ascending order:
1 5 10 12 23 34 45 67 89 90

=== Code Execution Successful ===
```

RESULT:-

Thus, the C program to sort a list of 10 numbers using Bubble Sort was successfully executed, and the numbers were arranged in ascending order.

6.2 Develop a Program to Perform addition of 3x3 matrix.

AIM: - To develop and execute a C program to perform addition of two 3×3 matrices and display the resultant matrix.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

A matrix is a rectangular arrangement of numbers in rows and columns.

Matrix addition is possible only when both matrices have the same order.

For two matrices **A** and **B** of order 3×3:

C[i][j] = A[i][j] + B[i][j]

where

- i → row index
- j → column index

Each element of the resultant matrix is obtained by adding corresponding elements of the two matrices.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare three 3×3 matrices.

Step 3: Read elements of the first matrix from the user.

Step 4: Read elements of the second matrix from the user.

Step 5: Add corresponding elements of both matrices.

Step 6: Store the result in the third matrix.

Step 7: Display the resultant matrix.

Step 8: Stop the program.

PROGRAM:-

```
#include <stdio.h>
int main()
{
   int a[3][3], b[3][3], c[3][3];
```

```
    int i, j;
    printf("Enter elements of first 3x3 matrix:\n");
    for(i = 0; i < 3; i++)
    {  for(j = 0; j < 3; j++)
       {   scanf("%d", &a[i][j]);  } }
    printf("Enter elements of second 3x3 matrix:\n");
    for(i = 0; i < 3; i++)
    { for(j = 0; j < 3; j++)
{  scanf("%d", &b[i][j]);  } }
    for(i = 0; i < 3; i++)
    {  for(j = 0; j < 3; j++)
       {  c[i][j] = a[i][j] + b[i][j]; } }
    printf("Resultant Matrix after Addition:\n");
    for(i = 0; i < 3; i++)
    { for(j = 0; j < 3; j++)
       { printf("%d\t", c[i][j]); }
       printf("\n");}
    return 0; }
```

OUTPUT:-

```
Enter elements of first 3x3 matrix:
1 2 3
4 5 6
7 8 9

Enter elements of second 3x3 matrix:
9 8 7
6 5 4
3 2 1

Resultant Matrix after Addition:
10  10  10
10  10  10
10  10  10


=== Code Execution Successful ===
```

RESULT:-

Thus, the C program to perform addition of two 3×3 matrices was successfully executed and the correct resultant matrix was obtained.

EXPERIMENT NO: 7

7.1 Develop Program to Create a structure called "library" to hold details of a book viz. accession number, title of the book, author name, price of the book, and flag indicating whether book is issued or not. Fetch some sample data and display the same.

AIM: - To develop a C program to create a structure called library to store book details such as accession number, title, author name, price, and a flag indicating whether the book is issued or not, fetch sample data, and display the same.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In C programming, a structure is a user-defined data type that allows grouping of variables of different data types under a single name.

Structures are useful for representing real-world entities like books in a library, where multiple attributes are associated with one entity.

In this program:

- A structure named library is defined.
- It contains:
  - Accession Number (integer)
  - Title of the book (string)
  - Author name (string)
  - Price (float)
  - Issue flag (integer: 1 = Issued, 0 = Not Issued)
- Sample data is stored in structure variables.
- The stored data is displayed using standard output.

PROCEDURE:-

Step 1: Start the program.

Step 2: Include required header files.

Step 3: Define a structure named library.

Step 4: Declare structure variables.

Step 5: Assign sample values to the structure members.

Step 6: Display the stored book details.

Step 7: Stop the program.

PROGRAM:-

```
 #include  <stdio.h>
#include  <string.h>
struct library
{ int  accession_number;
    char title[50];
    char  author[50];
    float price;
    int  issued;  };
int main()
{  struct library book;
    book.accession_number = 101;
    strcpy(book.title, "Programming in C");
    strcpy(book.author, "Dennis Ritchie");
    book.price = 450.50;
    book.issued = 1;
    printf("Library Book Details\n");
    printf("------------------------\n");
    printf("Accession Number : %d\n", book.accession_number);
    printf("Title            : %s\n", book.title);
    printf("Author             : %s\n", book.author);
    printf("Price            : Rs. %.2f\n", book.price);
    if (book.issued == 1)
       printf("Book Status      : Issued\n");
    else
       printf("Book Status      : Not Issued\n");
    return 0; }
```

OUTPUT:-

```
Library Book Details
--------------------
Accession Number : 101
Title            : Programming in C
Author           : Dennis Ritchie
Price            : Rs. 450.50
Book Status      : Issued


=== Code Execution Successful ===
```

RESULT:-

Thus, a C program using a structure named library was successfully developed to store and display book details such as accession number, title, author name, price, and issue status without any errors.

7.2 Develop and execute C Program to Add Two Distances given in kilometer-meter Using Structures

AIM: - To develop and execute a C program using structures to add two distances given in kilometers and meters and display the result in proper format.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In C programming, a structure is a user-defined data type that allows grouping of variables of different data types under a single name. Structures are useful when related data items need to be handled together.

In this program:

- A structure named Distance is used to store kilometers and meters.
- Two distance values are accepted from the user.
- The distances are added separately.
- If the meter value exceeds 1000 meters, it is converted into kilometers.

Conversion rule:

1000 meters = 1 kilometer

PROCEDURE:-

Step 1: Start the program.

Step 2: Define a structure to store kilometer and meter values.

Step 3: Declare structure variables for two distances and the result.

Step 4: Read input distances from the user.

Step 5: Add kilometers and meters separately.

Step 6: Convert meters to kilometers if meters ≥ 1000.

Step 7: Display the final distance.

Step 8: Stop the program.

PROGRAM:-

```
#include <stdio.h>
struct Distance {
```

```
    int km;
    int meter;};
int main() {
    struct Distance d1, d2, sum;
    printf("Enter first distance:\n");
    printf("Kilometers: ");
    scanf("%d", &d1.km);
    printf("Meters: ");
    scanf("%d", &d1.meter);
    printf("\nEnter second distance:\n");
    printf("Kilometers: ");
    scanf("%d", &d2.km);
    printf("Meters: ");
    scanf("%d", &d2.meter);
    sum.km = d1.km + d2.km;
    sum.meter = d1.meter + d2.meter;
    if (sum.meter >= 1000) {
        sum.km += sum.meter / 1000;
        sum.meter = sum.meter % 1000;  }
    printf("\nSum of distances = %d Kilometers %d Meters\n", sum.km, sum.meter);
    return 0; }
```

OUTPUT:-

```
Enter first distance:
Kilometers: 3
Meters: 750

Enter second distance:
Kilometers: 2
Meters: 500

Sum of distances = 6 Kilometers 250 Meters


=== Code Execution Successful ===
```

RESULT:-

The C program was successfully developed and executed.

It correctly adds two distances using structures, performs meter-to-kilometre conversion, and displays the final distance accurately.

EXPERIMENT NO: 8

8.1.1 Develop Program to demonstrate Use of all String handling functions.

AIM: - To develop and execute a C program to demonstrate the use of various string handling functions provided by the string.h header file.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In C programming, strings are arrays of characters terminated by a null character ('\0'). The string.h library provides predefined functions to perform operations such as copying, concatenation, comparison, length calculation, searching, and reversing strings.

Common String Handling Functions:

Function	Purpose
strlen()	Finds length of string
strcpy()	Copies one string to another
strcat()	Concatenates strings
strcmp()	Compares two strings
strrev()	Reverses string *(compiler dependent)*
strlwr()	Converts string to lowercase
strupr()	Converts string to uppercase
strchr()	Finds character in string
strstr()	Finds substring

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare character arrays to store strings.

Step 3: Accept input strings from the user.

Step 4: Apply string handling functions one by one.

Step 5: Display the results for each function.

Step 6: Stop the program.

PROGRAM:-

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[50], str2[50], str3[100];
    char *pos;
    int len, cmp;
    /* Input strings */
    printf("Enter first string: ");
    fgets(str1, sizeof(str1), stdin);
    str1[strcspn(str1, "\n")] = '\0';
    printf("Enter second string: ");
    fgets(str2, sizeof(str2), stdin);
    str2[strcspn(str2, "\n")] = '\0';
    /* strlen() */
    len = strlen(str1);
    printf("\nLength of first string: %d", len);
    /* strcpy() */
    strcpy(str3, str1);
    printf("\nCopied string (str3): %s", str3);
    /* strcat() */
    strcat(str3, str2);
    printf("\nConcatenated string (str3): %s", str3);
    /* strcmp() */
    cmp = strcmp(str1, str2);
    if (cmp == 0)
        printf("\nStrings are equal");
    else
        printf("\nStrings are not equal");
    /* strchr() */
    pos = strchr(str3, 'a');
    if (pos != NULL)
        printf("\nCharacter 'a' found in the string");
```

```
    else
        printf("\nCharacter 'a' not found in the string");
    /* strstr() */
    pos = strstr(str3, str2);
    if (pos != NULL)
        printf("\nSubstring found in the string");
    else
        printf("\nSubstring not found in the string");
    return 0;
}
```

OUTPUT:-

```
Enter first string: hello
Enter second string: world

Length of first string: 5
Copied string (str3): hello
Concatenated string (str3): helloworld
Strings are not equal
Character 'a' not found in the string
Substring found in the string

=== Code Execution Successful ===
```

RESULT:-

Thus, the program was successfully executed and the working of string handling functions in C was demonstrated.

8.1.2 Develop Program to demonstrate Use of few Mathematical functions.

AIM: - To develop a C program to demonstrate the use of few mathematical functions such as square root, power, floor, ceil, and trigonometric functions using the math.h library.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

C provides a rich set of mathematical functions through the header file math.h. These functions are used to perform complex mathematical operations easily.

Some commonly used mathematical functions are:

- sqrt(x) – Calculates square root
- pow(x, y) – Calculates x raised to power y
- floor(x) – Returns the largest integer less than or equal to x
- ceil(x) – Returns the smallest integer greater than or equal to x
- sin(x), cos(x) – Trigonometric functions (angle in radians)

To use these functions, the program must include the math.h header file.

PROCEDURE:-

Step 1: Start the program.

Step 2: Include required header files stdio.h and math.h.

Step 3: Declare required variables.

Step 4: Read input values from the user.

Step 5: Apply mathematical functions using math.h.

Step 6: Display the results.

Step 7: Stop the program.

PROGRAM:-

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

```
double num, power, angle;
printf("Enter a number: ");
scanf("%lf", &num);
printf("Enter power value: ");
scanf("%lf", &power);
printf("Enter angle in radians: ");
scanf("%lf", &angle);
printf("\nSquare Root of %.2lf = %.2lf", num, sqrt(num));
printf("\n%.2lf raised to power %.2lf = %.2lf", num, power, pow(num, power));
printf("\nFloor value of %.2lf = %.2lf", num, floor(num));
printf("\nCeil value of %.2lf = %.2lf", num, ceil(num));
printf("\nSine of %.2lf = %.2lf", angle, sin(angle));
printf("\nCosine of %.2lf = %.2lf", angle, cos(angle));
return 0; }
```

OUTPUT:-

```
Enter a number: 9
Enter power value: 2
Enter angle in radians: 1.57

Square Root of 9.00 = 3.00
9.00 raised to power 2.00 = 81.00
Floor value of 9.00 = 9.00
Ceil value of 9.00 = 9.00
Sine of 1.57 = 1.00
Cosine of 1.57 = 0.00

=== Code Execution Successful ===
```

RESULT:-

Thus, the C program to demonstrate the use of few mathematical functions using math.h was successfully executed and the output was obtained correctly.

8.1.3 Develop Program to demonstrate few other miscellaneous functions.

AIM: - To develop a C program to demonstrate the use of miscellaneous functions such as rand(), srand(), time(), abs(), and toupper().

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

C provides several miscellaneous library functions to perform common tasks:

1. rand()
    - Generates a pseudo-random number.
    - Defined in <stdlib.h>.
2. srand()
    - Seeds the random number generator.
    - Usually seeded using time() to get different random values.
    - Defined in <stdlib.h>.
3. time()
    - Returns the current system time.
    - Used to generate different random values each time the program runs.
    - Defined in <time.h>.
4. abs()
    - Returns the absolute value of an integer.
    - Defined in <stdlib.h>.
5. toupper()
    - Converts a lowercase character to uppercase.
    - Defined in <ctype.h>.

PROCEDURE:-

Step 1: Start the program.

Step 2: Include required header files.

Step 3: Seed the random number generator using srand(time(NULL)).

Step 4: Generate a random number using rand().

Step 5: Find the absolute value of a number using abs().

Step 6: Convert a lowercase character to uppercase using toupper().

Step 7: Display the results.

Step 8: Stop the program.

PROGRAM:-

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>
int main()
{
   int randomNumber;
   int number = -25;
   char ch = 'a';
   srand(time(NULL));
   randomNumber = rand();
   printf("Random Number: %d\n", randomNumber);
   printf("Absolute value of %d is %d\n", number, abs(number));
   printf("Uppercase of '%c' is '%c'\n", ch, toupper(ch));
   return 0; }
```

OUTPUT:-

```
Random Number: 1243216361
Absolute value of -25 is 25
Uppercase of 'a' is 'A'


=== Code Execution Successful ===
```

RESULT:-

Thus, the C program was successfully developed and executed to demonstrate the use of miscellaneous functions rand(), srand(), time(), abs(), and toupper().

8.2.1 Develop a Program to Create a function to find GCD of given number. Call this function in a program.

AIM: - To develop a C program to create a user-defined function to find the Greatest Common Divisor (GCD) of two given numbers and call this function in the main program.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

The Greatest Common Divisor (GCD) of two numbers is the largest positive integer that divides both numbers without leaving a remainder.

In this program, a user-defined function is used to calculate the GCD using Euclid's Algorithm. According to this algorithm:

- GCD of two numbers a and b is calculated by repeatedly replacing the larger number with the remainder of the division until the remainder becomes zero.
- The non-zero number at that stage is the GCD.

Using functions improves modularity, readability, and reusability of the program.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare required variables.

Step 3: Create a user-defined function gcd() to compute GCD.

Step 4: Read two numbers from the user.

Step 5: Call the gcd() function from the main() function.

Step 6: Display the GCD of the given numbers.

Step 7: Stop the program.

PROGRAM:-

```
#include  <stdio.h>
int gcd(int a, int b);
int main(){
   int num1, num2, result;
```

```
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    result = gcd(num1, num2);
    printf("GCD of %d and %d is %d", num1, num2, result);
    return 0;}
int gcd(int a, int b){
    int temp;
    while (b != 0) {
        temp = b;
        b = a % b;
        a = temp; }
    return a; }
```

OUTPUT:-

```
Enter two numbers: 24 36
GCD of 24 and 36 is 12

=== Code Execution Successful ===
```

RESULT:-

Thus, a C program to find the GCD of two numbers using a user-defined function was successfully developed and executed.

8.2.2 Develop a Program to Find Factorial of given number using recursion.

AIM: - To develop a C program to find the factorial of a given number using recursion.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

Factorial of a non-negative integer n is defined as:

$n!=n\times(n-1)\times(n-2)\times...\times1$

By definition:

- 0! = 1
- 1! = 1

Recursion is a technique in which a function calls itself to solve a problem.

A recursive function must have:

1. Base condition – stops recursion
2. Recursive call – function calling itself

Recursive definition of factorial:

- factorial(0) = 1
- factorial(n) = n × factorial(n-1)

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare a recursive function to calculate factorial.

Step 3: Accept an integer value from the user.

Step 4: Call the recursive function.

Step 5: Display the factorial of the given number.

Step 6: Stop the program.

PROGRAM:-

```
#include <stdio.h>

int factorial(int n);

int main(){
```

```
    int num, result;
    printf("Enter a number: ");
    scanf("%d", &num);
    result = factorial(num);
    printf("Factorial of %d is %d", num, result);
    return 0; }
int factorial(int n){
    if (n == 0)
        return 1;
    else
        return n * factorial(n - 1); }
```

OUTPUT:-

```
Enter a number: 5
Factorial of 5 is 120

=== Code Execution Successful ===
```

RESULT:-

The C program to find the factorial of a given number using recursion was successfully executed and the correct output was obtained.

EXPERIMENT NO: 9

9.1 Develop a Program to Print values of variables and their addresses.

AIM: - To develop a C program to print the values of variables and their memory addresses.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In C programming, every variable is stored in a specific memory location.

The address-of operator (&) is used to find the memory address of a variable.

Printing variable values and their addresses helps in understanding:

Memory allocation

- Pointer concepts
- How data is stored in RAM

The printf() function with format specifier %p is used to display addresses.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare variables of different data types

Step 3: Assign values to the variables.

Step 4: Use printf() to print variable values.

Step 5: Use the address-of operator (&) to print memory addresses.

Step 6: Stop the program.

PROGRAM:-

```
#include <stdio.h>
int main()
{
   int a = 10;
   float b = 25.5;
   char c = 'A';
   printf("Value of a = %d\n", a);
```

```
printf("Address of a = %p\n\n", &a);
printf("Value of b = %.2f\n", b);
printf("Address of b = %p\n\n", &b);
printf("Value of c = %c\n", c);
printf("Address of c = %p\n", &c);
return 0; }
```

OUTPUT:-

```
Value of a = 10
Address of a = 0x7ffeb7ed929c

Value of b = 25.50
Address of b = 0x7ffeb7ed9298

Value of c = A
Address of c = 0x7ffeb7ed9297


=== Code Execution Successful ===
```

RESULT:-

Thus, the C program to print the values of variables and their memory addresses was executed successfully.

9.2 Develop a Program to Find sum of all elements stored in given array using pointers.

AIM: - To develop a C program to find the sum of all elements stored in a given array using pointers.

SOFTWARE REQURIED:-

1. Turbo C / GCC Compiler
2. Code::Blocks / Dev-C++ / Visual Studio Code
3. Windows / Linux OS

THEORY:-

In C programming, a pointer is a variable that stores the address of another variable.

Instead of accessing array elements using array indexing, we can use pointers to traverse the array. The name of an array acts as a pointer to the first element of the array. By incrementing the pointer, we can access successive elements of the array. This approach improves understanding of memory management and pointer arithmetic.

PROCEDURE:-

Step 1: Start the program.

Step 2: Declare an integer array and required variables.

Step 3: Read the number of elements.

Step 4: Input the array elements.

Step 5: Assign the base address of the array to a pointer.

Step 6: Use the pointer to access each element and calculate the sum.

Step 7: Display the sum of array elements.

Step 8: Stop the program.

PROGRAM:-

```
#include <stdio.h>
int main()
{
   int arr[100], n, i, sum = 0;
   int *ptr;
   printf("Enter number of elements: ");
   scanf("%d", &n);
```

```
    printf("Enter array elements:\n");
    for(i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    ptr = arr;  // pointer points to base address of array
    for(i = 0; i < n; i++)
    {
        sum = sum + *(ptr + i);
    }
    printf("Sum of array elements = %d\n", sum);
    return 0;
}
```

OUTPUT:-

```
Enter number of elements: 5
Enter array elements:
10
20
30
40
50
Sum of array elements = 150


=== Code Execution Successful ===
```

RESULT:-

Thus, the C program to find the sum of all elements of an array using pointers was successfully executed and the correct result was obtained.