PESCE FOSS CLUB
www.pescefoss.in

Principles of Programming using C Lab

LAB MANUAL

Dr. Bramesh S M
Mentor, PESCE FOSS

This document has been typeset using the LaTeX document preparation system.

Contents

Principles of Programming using C (Integrated)
Common to all Branches

Sub Code: P24ESCS103/203 **CIE Marks : 50**
Hrs/ Week: 2:0:2:0 **Exam Hours : 3 + 2**
Total Hrs.: 40 **Exam Marks : 50**

Practical Examination Procedure:

- All experiments are to be included for practical examination.
- Students are allowed to pick one experiment from the lot.
- Strictly follow the instructions as printed on the answer script for breakup of marks.
- **Change of experiment is allowed only once and 15% Marks should be deducted from the procedure part.**

Part I

PART A

0.1 Syllabus

List of Experiments

1. Simulation of a Simple Calculator.
2. Compute the roots of a quadratic equation by accepting the co-efficients a, b, and c. Print appropriate messages based on the nature of roots.
3. An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit:for the next 100 units 90 paise per unit: beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs.100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amountis charged. Write a program to read the name of the user, number of units consumed and print out the charges.
4. Write a C Program to display the following by reading the number of rows as input,

   1
   1 2 1
   1 2 3 2 1
   1 2 4 3 2 1

   - - - - - - - - - -
   n^{th} row
5. Implement Binary Search on Integers.
6. Implement Matrix multiplication and validate the rules of multiplication.
7. Compute sin(x)/cos(x) using Taylor series approximation. Compare your result with the built-in library function.Print both the results with appropriate inferences.
8. Sort the given set of N numbers using Bubble sort.
9. Write functions to implement string operations such as compare, concatenate, and find string length. Use theparameter passing techniques.
10. Implement structures to read, write and compute average- marks of the students, list the students scoring aboveand below the average marks for a class of N students.
11. Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored inan array of N real numbers.
12. Write a C program to copy a text file to another, read both the input file name and target file name.

Part II

PART B

Problem Solving in C

Implement the programs with GNU / LINUX platform using appropriate C compiler.

Experiment 1

Simple Calculator

Design and develop a simple calculator.

C Code

```
/*****************************************************************************
*File         : 1.c
*Description  : Program to simulate a simple calculator
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

#include<stdio.h>

/*****************************************************************************
*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/
int main()
{
 char op;
 double first, second;
 printf("Enter an operator (+, -, *, /): \n");
 scanf("%c", &op);
 printf("Enter two operands: \n");
 scanf("%lf %lf", &first, &second);
 switch (op)
 {
  case '+':
           printf("%.1lf + %.1lf = %.1lf \n", first, second, first + second);
           break;
  case '-':
           printf("%.1lf - %.1lf = %.1lf \n", first, second, first - second);
           break;
  case '*':
           printf("%.1lf * %.1lf = %.1lf \n", first, second, first * second);
           break;
  case '/':
           printf("%.1lf / %.1lf = %.1lf \n", first, second, first / second);
           break;
  default:
           printf("Error! operator is not correct \n");
 }
 return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 1.c
$ ./a.out

```
Enter an operator (+, -, *, /):
@
Enter two operands:
1
2

Error! operator is not correct
```

$ ./a.out

```
Enter an operator (+, -, *, /):
+
Enter two operands:
12
2

12.0 + 2.0 = 14.0
```

Experiment 2

Quadratic Equation

Compute the roots of a quadratic equation by accepting the co-efficients a, b, and c. Print appropriate messages based on the nature of roots.

C Code

```
/*****************************************************************************
*File         : 2.c
*Description  : Program to compute the roots of a quadratic equation
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

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

/*****************************************************************************
*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/
int main()
{
    double a, b, c, discriminant, root1, root2, realPart, imagPart;

    // Accept coefficients from the user
    printf("Enter coefficients a, b, and c: ");
    scanf("%lf %lf %lf", &a, &b, &c);

    // Check if it's a quadratic equation
    if (a == 0)
    {
        printf("This is not a quadratic equation.\n");
        return 0;
    }

    // Compute the discriminant
    discriminant = b * b - 4 * a * c;

    // Check the nature of the roots
    if (discriminant > 0)
    {
        // Two distinct real roots
        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("Roots are real and distinct: %.2lf and %.2lf\n", root1, root2);
     }
```

```
    else if (discriminant == 0)
    {
       // One real root (double root)
       root1 = -b / (2 * a);
       printf("Roots are real and equal: %.2lf\n", root1);
    }
    else
    {
       // Complex roots
       realPart = -b / (2 * a);
       imagPart = sqrt(-discriminant) / (2 * a);
       printf("Roots are complex: %.2lf + %.2lfi and %.2lf - %.2lfi\n", realPart, imagPart, realPart, i
    }

   return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 2.c -lm
$ ./a.out

```
Enter coefficients a, b, and c: 1 -3 2

Roots are real and distinct: 2.00 and 1.00
```

$ ./a.out

```
Enter coefficients a, b, and c: 1 -2 1

Roots are real and equal: 1.00
```

$ ./a.out

```
Enter coefficients a, b, and c: 1 2 5

Roots are complex: -1.00 + 2.00i and -1.00 - 2.00i
```

$ ./a.out

```
Enter coefficients a, b, and c: 0 1 1

This is not a quadratic equation.
```

Experiment 3

Electricity Billing

An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units rupees 1 per unit. All users are charged a minimum of rupees 100 as a meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15 % of the total amount is charged. Write a program to read the name of the user, the number of units consumed, and print out the charges.

C Code

```
/*******************************************************************************
*File         : 3.c
*Description  : Program for Electricity Billing
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*******************************************************************************/

#include<stdio.h>

/*******************************************************************************
*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*******************************************************************************/
int main()
{
    char name[50];
    int units;
    float charge = 0.0, totalCharge = 0.0, surcharge = 0.0;
    const float meterCharge = 100.0;

    printf("Enter the user's name: ");
    scanf("%s", name);

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

    // Calculate base charge
    if (units <= 200) {
        charge = units * 0.80;
    } else if (units <= 300) {
        charge = (200 * 0.80) + ((units - 200) * 0.90);
    } else {
        charge = (200 * 0.80) + (100 * 0.90) + ((units - 300) * 1.00);
    }

    // Add fixed meter charge
    totalCharge = charge + meterCharge;
```

```
    // Apply 15% surcharge if total is over 400
    if (totalCharge > 400) {
        surcharge = totalCharge * 0.15;
        totalCharge += surcharge;
    }

    // Print the bill
    printf("\n------------------------------\n");
    printf("      ELECTRICITY BILL        \n");
    printf("------------------------------\n");
    printf("Name            : %s\n", name);
    printf("Units Consumed  : %d\n", units);
    printf("Energy Charge   : Rs. %.2f\n", charge);
    printf("Meter Charge    : Rs. %.2f\n", meterCharge);
    if (surcharge > 0) {
        printf("Surcharge (15%%) : Rs. %.2f\n", surcharge);
    }
    printf("------------------------------\n");
    printf("Total Amount    : Rs. %.2f\n", totalCharge);
    printf("------------------------------\n");

    return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 3.c
$ ./a.out

```
Enter the user's name: ISE
Enter the number of units consumed: 350

------------------------------
      ELECTRICITY BILL
------------------------------
Name            : smb
Units Consumed  : 350
Energy Charge   : Rs. 300.00
Meter Charge    : Rs. 100.00
------------------------------
Total Amount    : Rs. 400.00
------------------------------
```

$ ./a.out

```
Enter the user's name: ISE
Enter the number of units consumed: 500

------------------------------
      ELECTRICITY BILL
------------------------------
Name            : ISE
Units Consumed  : 500
Energy Charge   : Rs. 450.00
Meter Charge    : Rs. 100.00
Surcharge (15%) : Rs. 82.50
------------------------------
Total Amount    : Rs. 632.50
------------------------------
```

Experiment 4

Pyramid Pattern

Write a C Program to display the following by reading the number of rows as input,

1
1 2 1
1 2 3 2 1
1 2 4 3 2 1

----
n^{th} **row**

C Code

```
/*****************************************************************************
*File         : 4.c
*Description  : Program to Print a Symmetrical Number Pyramid Pattern
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

#include<stdio.h>

/*****************************************************************************
*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/
int main()
{

 int rows, i, j;

 // Read number of rows
 printf("Enter the number of rows: ");
 scanf("%d", &rows);

 for(i = 1; i <= rows; i++)
 {
    // Print leading spaces
    for(j = 1; j <= rows-i; j++)
    {
      printf(" ");
    }
    // Print increasing numbers
    for(j = 1; j <= i; j++)
    {
      printf("%d",j);
    }
```

```
    // Print decreasing numbers
    for( j= i-1;j >= 1; j--)
    {
      printf("%d",j);
    }

    // Move to the next line
    printf("\n");
  }
  return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 4.c
$ ./a.out

```
Enter the number of rows: 4
   1
  121
 12321
1234321
```

$ ./a.out

```
Enter the number of rows: 5
    1
   121
  12321
 1234321
123454321
```

Experiment 5

Binary Search

Implement Binary Search on Integers

C Code

```
/*****************************************************************************
*File         : 5.c
*Description  : Program to implement binary search
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

#include<stdio.h>

/*****************************************************************************
*Function         :        binarySearch
*Input parameters :        array, key, low, high
*RETURNS          :        Index of the key element in the array on success

*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/

// Function declaration for binary search
int binarySearch(int array[], int key, int low, int high);

int main()
{
  // Initialize a sorted array
  int array[] = {3, 4, 5, 6, 7, 8, 9};

  // Calculate number of elements in the array
  int n = sizeof(array) / sizeof(array[0]);

  // Element to search for in the array
        int key = 3;

  // Call binarySearch and store result
  int result = binarySearch(array, key, 0, n-1);

  // Print result based on whether key was found
  if (result == -1)
  {
    printf("Not found");
  }
  else
```

```
  {
    printf("Element is found at index %d", result);
  }
  return 0;
}

// Function to perform binary search
int binarySearch(int array[], int key, int low, int high)
{
   // Repeat untill there are no elements in the array
   while (low <= high)
   {
     // Calculate middle index
     int mid = low + (high - low) / 2;

     // If the key is found at mid
     if (array[mid] == key)
     {
        return mid;  // Return Index
     }

    // If key is greater, ignore the left half
    if (array[mid] < key)
    {
       low = mid + 1;
    }
    // If key is smaller, ignore the right half
    else
    {
      high = mid - 1;
    }
  }
  // Key not found
  return -1;
}
```

Output

Run the following commands in your terminal:
$ gcc 5.c
$ ./a.out

```
Element is found at index 0
```

Experiment 6

Matrix Multiplication

Implement Matrix multiplication and validate the rules of multiplication.

C Code

```
/*****************************************************************************
*File         : 6.c
*Description  : Program to implement matrix multiplication
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

#include<stdio.h>

/*****************************************************************************
*Function         :        multiplyMatrices
*Input parameters :        firstMatrix, secondMatrix, result, r1, c1, c2
*RETURNS          :        void

*Function         :        displayMatrix
*Input parameters :        matrix, rows, cols
*RETURNS          :        void

*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/

// Function to multiply matrices
void multiplyMatrices(int firstMatrix[10][10], int secondMatrix[10][10], int result[10][10], int r1, int
{
    for (int i = 0; i < r1; i++)
    {
        for (int j = 0; j < c2; j++)
        {
            result[i][j] = 0;
            for (int k = 0; k < c1; k++)
            {
                result[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
            }
        }
    }
}
```

```
// Function to display a matrix
void displayMatrix(int matrix[10][10], int rows, int cols)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}

int main()
{
    int firstMatrix[10][10], secondMatrix[10][10], result[10][10];
    int r1, c1, r2, c2;

    // Input dimensions
    printf("Enter rows and columns of first matrix: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter rows and columns of second matrix: ");
    scanf("%d %d", &r2, &c2);

    // Validate matrix multiplication condition
    if (c1 != r2) {
        printf("Matrix multiplication not possible. Columns of first matrix must equal rows of second ma
        return 1;
    }

    // Input matrices
    printf("Enter elements of first matrix:\n");
    for (int i = 0; i < r1; i++)
        for (int j = 0; j < c1; j++)
            scanf("%d", &firstMatrix[i][j]);

    printf("Enter elements of second matrix:\n");
    for (int i = 0; i < r2; i++)
        for (int j = 0; j < c2; j++)
            scanf("%d", &secondMatrix[i][j]);

    // Multiply matrices
    multiplyMatrices(firstMatrix, secondMatrix, result, r1, c1, c2);

    // Display result
    printf("Resultant matrix after multiplication:\n");
    displayMatrix(result, r1, c2);

    return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 6.c
$ ./a.out

```
Enter rows and columns of first matrix: 2 2

Enter rows and columns of second matrix: 2 2

Enter elements of first matrix:
1 2
1 2

Enter elements of second matrix:
1 12
1 2

Resultant matrix after multiplication:
3       16
3       16
```

$ ./a.out

```
Enter rows and columns of first matrix: 3 3

Enter rows and columns of second matrix: 4 3

Matrix multiplication not possible. Columns of first matrix must equal rows of second matrix.
```

Experiment 7

Taylor Series Approximation

Compute sin(x) / cos(x) using Taylor series approximation. Compare your result with the built-in library function.Print both the results with appropriate inferences.

$$Taylor Series for sin(x) : \sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

C Code

```
/*****************************************************************************
*File         : 7.c
*Description  : Program to implement Taylor series approximation
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/
#include<stdio.h>
#include<math.h>
/*****************************************************************************
*Function         :        toRadians
*Input parameters :        degrees
*RETURNS          :        radians

*Function         :        factorial
*Input parameters :        n
*RETURNS          :        factorial of n

*Function         :        taylorSin
*Input parameters :        x, terms
*RETURNS          :        sin(x)

*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/
// Function to convert degrees to radians
double toRadians(double degrees)
{
    return degrees * (M_PI / 180.0);
}

// Function to calculate factorial
long factorial(int n)
{
    long fact = 1;
    for (int i = 2; i <= n; i++)
        fact *= i;
    return fact;
}
```

```
// Function to calculate sin(x) using Taylor series
double taylorSin(double x, int terms)
{
    double result = 0.0;
    int sign = 1;
    for (int i = 0; i < terms; i++)
    {
        int termPower = 2 * i + 1;
        double term = sign * pow(x, termPower) / factorial(termPower);
        result += term;
        sign *= -1; // alternate signs
    }
    return result;
}

int main()
{
    double degrees, radians;
    int terms = 10;

    printf("Enter angle in degrees: ");
    scanf("%lf", &degrees);

    radians = toRadians(degrees);

    double approx = taylorSin(radians, terms);
    double actual = sin(radians);

    printf("\nUsing Taylor Series Approximation (terms = %d): %.10lf", terms, approx);
    printf("\nUsing built-in sin() function: %.10lf", actual);
    printf("\nDifference: %.10lf\n", fabs(approx - actual));

    return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 7.c -lm
$ ./a.out

```
Enter angle in degrees: 30

Using Taylor Series Approximation (terms = 10): 0.5000000000
Using built-in sin() function: 0.5000000000
Difference: 0.0000000000
```

$ ./a.out

```
Enter angle in degrees: 90

Using Taylor Series Approximation (terms = 10): 1.0000000000
Using built-in sin() function: 1.0000000000
Difference: 0.0000000000
```

Experiment 8

Bubble Sort

Sort the given set of N numbers using Bubble sort

C Code

```
/*****************************************************************************
*File         : 8.c
*Description  : Program to implement Bubble sort
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

#include<stdio.h>

/*****************************************************************************
*Function         :        bubbleSort
*Input parameters :        arr, n
*RETURNS          :        void

*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/

// Function to perform Bubble Sort
void bubbleSort(int arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
    {
        // Last i elements are already sorted
        for (int j = 0; j < n - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                // Swap arr[j] and arr[j+1]
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

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

```
    // Ask user for number of elements
    printf("Enter number of elements: ");
    scanf("%d", &n);

    // Input array elements
    printf("Enter %d integers:\n", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    // Call function to sort the array
    bubbleSort(arr, n);

    // Display sorted array
    printf("Sorted array:\n");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);

    printf("\n");

    return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 8.c
$ ./a.out

```
Enter number of elements: 6
Enter 6 integers:
6
5
4
3
2
1
Sorted array:
1 2 3 4 5 6
```

$ ./a.out

```
Enter number of elements: 3
Enter 3 integers:
10
5
7
Sorted array:
5 7 10
```

Experiment 9

String Operations

Write functions to implement string operations such as compare, concatenate, and find string length. Use the parameter passing techniques.

C Code

```
/*****************************************************************************
*File         : 9.c
*Description  : Program to implement String operations
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

#include<stdio.h>

/*****************************************************************************
*Function         :        stringLength
*Input parameters :        string
*RETURNS          :        length of the string (int)

*Function         :        stringCompare
*Input parameters :        two strings
*RETURNS          :        Equal or Not Equal (int)

*Function         :        stringConcatenate
*Input parameters :        two strings
*RETURNS          :        void

*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/

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

// Function to compare two strings
int stringCompare(char str1[], char str2[])
{
```

```
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0')
    {
        if (str1[i] != str2[i])
            return 0; // Strings are not equal
        i++;
    }
    // If both strings ended at the same time, they're equal
    return (str1[i] == '\0' && str2[i] == '\0');
}

// Function to concatenate two strings
void stringConcatenate(char str1[], char str2[])
{
    int i = 0, j = 0;

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

    // Append str2 to the end of str1
    while (str2[j] != '\0')
    {
        str1[i] = str2[j];
        i++;
        j++;
    }

    // Null-terminate the result
    str1[i] = '\0';
}

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

    // Input strings
    printf("Enter first string: ");
    scanf("%s", str1); // Reads string until space
    printf("Enter second string: ");
    scanf("%s", str2);

    // Compare
    if (stringCompare(str1, str2))
        printf("\nStrings are equal.\n");
    else
        printf("\nStrings are NOT equal.\n");

    // Lengths
    printf("Length of first string: %d\n", stringLength(str1));
    printf("Length of second string: %d\n", stringLength(str2));

    // Concatenate
    stringConcatenate(str1, str2);
    printf("After concatenation: %s\n", str1);

    return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 9.c
$ ./a.out

```
Enter first string: hello
Enter second string: ise

Strings are NOT equal.
Length of first string: 5
Length of second string: 3
After concatenation: helloise
```

$ ./a.out

```
Enter first string: ise
Enter second string: cse

Strings are NOT equal.
Length of first string: 3
Length of second string: 3
After concatenation: isecse
```

Experiment 10

Student Records

Implement structures to read, write and compute average - marks of the students, list the students scoring above and below the average marks for a class of N students

C Code

```
/*****************************************************************************
*File         : 10.c
*Description  : Program to implement Student Records
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

#include<stdio.h>
#define MAX_STUDENTS 10
/*****************************************************************************
*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/

// Structure to store student data
struct Student
{
    char name[50];
    float marks;
};

int main()
{
    struct Student students[MAX_STUDENTS];
    int n;
    float sum = 0.0, avg;

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

    // Input student details
    for (int i = 0; i < n; i++)
    {
        printf("Enter name of student %d: ", i + 1);
        scanf("%s", students[i].name);
        printf("Enter marks of %s: ", students[i].name);
        scanf("%f", &students[i].marks);
        sum += students[i].marks;
    }
```

```
    // Compute average
    avg = sum / n;
    printf("\nAverage Marks = %.2f\n", avg);

    // List students scoring above average
    printf("\nStudents scoring above average:\n");
    for (int i = 0; i < n; i++)
    {
        if (students[i].marks > avg)
        {
            printf("%s - %.2f\n", students[i].name, students[i].marks);
        }
    }

    // List students scoring below average
    printf("\nStudents scoring below average:\n");
    for (int i = 0; i < n; i++)
    {
        if (students[i].marks < avg)
        {
            printf("%s - %.2f\n", students[i].name, students[i].marks);
        }
    }

    return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 10.c
$ ./a.out

```
Enter number of students: 3
Enter name of student 1: Ram
Enter marks of Ram: 50
Enter name of student 2: Raj
Enter marks of Raj: 60
Enter name of student 3: Ravi
Enter marks of Ravi: 30

Average Marks = 46.67

Students scoring above average:
Ram - 50.00
Raj - 60.00

Students scoring below average:
Ravi - 30.00
```

Experiment 11

Pointers and Arrays

Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers

C Code

```
/*****************************************************************************
*File         : 11.c
*Description  : Program to implement Pointers and Arrays
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

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

#define MAX_SIZE 10

/*****************************************************************************
*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/

int main()
{
    float arr[MAX_SIZE], sum = 0.0, mean, std_dev = 0.0;
    int n;
    float *ptr;

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

    // Input elements
    printf("Enter %d real numbers:\n", n);
    for (ptr = arr; ptr < arr + n; ptr++)
    {
        scanf("%f", ptr);
        sum += *ptr;
    }

    // Compute mean
    mean = sum / n;

    // Compute standard deviation
    for (ptr = arr; ptr < arr + n; ptr++)
```

```
    {
        std_dev += pow((*ptr - mean), 2);
    }
    std_dev = sqrt(std_dev / n);

    // Display results
    printf("\nSum = %.2f", sum);
    printf("\nMean = %.2f", mean);
    printf("\nStandard Deviation = %.2f\n", std_dev);

    return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 11.c -lm
$ ./a.out

```
Enter number of elements: 4
Enter 4 real numbers:
1
2
3
4

Sum = 10.00
Mean = 2.50
Standard Deviation = 1.12
```

$ ./a.out

```
Enter number of elements: 5
Enter 5 real numbers:
10
20
30
40
50

Sum = 150.00
Mean = 30.00
Standard Deviation = 14.14
```

Experiment 12

File Management

Write a C program to copy a text file to another, read both the input file name and target file name.

C Code

```
/*****************************************************************************
*File         : 12.c
*Description  : Program to implement Files
*Author       : Dr. Bramesh S M
*Compiler     : gcc 13.3.0 compiler, Ubuntu 24.04.01
*****************************************************************************/

#include<stdio.h>

/*****************************************************************************
*Function         :        main
*Input parameters :        no parameters
*RETURNS          :        0 on success
*****************************************************************************/
int main()
 {
    char sourceFile[100], targetFile[100];
    FILE *src, *tgt;
    char ch;

    // Input file names
    printf("Enter source file name: ");
    scanf("%s", sourceFile);

    printf("Enter target file name: ");
    scanf("%s", targetFile);

    // Open source file for reading
    src = fopen(sourceFile, "r");
    if (src == NULL) {
        printf("Cannot open source file.\n");
        return 1;
    }

    // Open target file for writing
    tgt = fopen(targetFile, "w");
    if (tgt == NULL) {
        printf("Cannot open target file.\n");
        fclose(src);
        return 1;
    }
```

```
    // Copy content character by character
    while ((ch = fgetc(src)) != EOF) {
        fputc(ch, tgt);
    }

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

    // Close files
    fclose(src);
    fclose(tgt);

    return 0;
}
```

Output

Run the following commands in your terminal:
$ gcc 12.c
$ ./a.out

```
Enter source file name: source.txt
Enter target file name: destination.txt
File copied successfully.
```

$ ./a.out

```
Enter source file name: test.txt
Enter target file name: destination.txt
Cannot open source file.
```