BMSIT & M, Bengaluru -560119

BPOP13/23 – C Lab Manual – 1st/2nd Sem CSE

Dr. Shankar R

Assistant Professor

CSE

BMSIT&M

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resource for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

PROGRAM OUTCOMES (POs)

1. *Apply knowledge of mathematics, science, Engineering fundamentals and computing skills to solve IT related engineering problems.*
2. *Identify, formulate, research literature and analyze complex computer science engineering problems.*
3. *Design software / hardware solutions for complex IT problems to uplift the societal status of common man.*
4. *Function effectively as an individual and as a member or leader in diverse teams / multidisciplinary teams.*
5. *Apply ethical principles in responsible way to follow the IT norms and cyber ethics.*
6. *Communicate effectively on complex engineering activities with the engineering community and the society for the effective presentation and / or report generation.*
7. *Use research-based knowledge and methods including design of software and or hardware experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.*
8. *Create/select an appropriate IT tool to model, implement and automate the complex computational system.*
9. *Apply reasoning informed by the contextual knowledge to assess societal, cultural, environmental issues and the consequent responsibilities relevant to the IT practice.*
10. *Understand the impact of the IT solutions to demonstrate the knowledge of, and need for sustainable IT development.*
11. *Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.*
12. *Demonstrate the knowledge of computing / managerial principles to solve and manage IT projects.*

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Lead a successful career by designing, analysing and solving various problems in the field of Computer Science & Engineering.

PEO 2: Pursue higher studies for enduring edification.

PEO 3: Exhibit professional and team building attitude along with effective communication.

PEO 4: Identify and provide solutions for sustainable environmental development.

PROGRAM SPECIFIC OBJECTIVES (PSOs)

PEO 1: Analyze the problem and identify computing requirements appropriate to its solution.

PEO 2: Apply design and development principles in the construction of software systems of varying complexity.

COURSE OBJECTIVES

This course will enable students to:

- Gain proficiency in the syntax and semantics of the C programming language.
- Understand the structure and components of a C program, including functions, data types, operators, and control & Looping structures.
- Learn to analyze problems, design algorithms, and implement solutions using C.
- Apply C programming skills to real-world problems and projects.

COURSE OUTCOMES (COS)

The students should be able to:

CO1:	Apply the concepts of Algorithm, flowchart and basic C constructs to solve the problems (L3)
CO2:	Make use of arrays & strings with modular programming concepts to solve problems. (L3)
CO3:	Apply the concepts of Pointers and Structures for real world scenarios. (L3)
CO4:	Develop C programs to solve practical problems, employing core programming principles and debugging techniques. (L3)
CO5:	Analyse the code snippets to identify the errors and predict the output. (L4)

List of Programs

Program Number	Questions
1	Compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.
2	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 amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.
3	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 3 4 3 2 1 1 2 3 2 1 1 2 1 1
4	Implement Binary Search on Integers
5	Implement Matrix multiplication and validate the rules of multiplication
6	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.
7	Using functions sort the given set of N numbers (Bubble sort).
8	Write user defined functions to implement string operations such as compare, concatenate, and find string length. Use the parameter passing techniques.
9	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.
10	Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.

1. Compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.

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

int main()
{
    float a, b, c;
    float discriminant, root1, root2;

    // Accept coefficients
    printf("Enter coefficients a, b and c of the quadratic equation
(ax^2 + bx + c = 0):\n");
    scanf("%f %f %f", &a, &b, &c);

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

    // Compute roots based on discriminant value
    if (discriminant > 0)
    {
        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("The equation has two real and distinct roots:\n");
        printf("Root 1 = %.2f\n", root1);
        printf("Root 2 = %.2f\n", root2);
    }
    else if (discriminant == 0)
    {
        root1 = -b / (2 * a);
        root2 = -b / (2 * a);
        printf("The equation has two real and equal roots:\n");
        printf("Root 1 = %.2f\n", root1);
        printf("Root 2 = %.2f\n", root2);
    }
    else
    {
        float realPart = -b / (2 * a);
        float imaginaryPart = sqrt(-discriminant) / (2 * a);
        printf("The equation has two complex roots:\n");
        printf("Root 1 = %.2f + %.2fi\n", realPart, imaginaryPart);
        printf("Root 2 = %.2f - %.2fi\n", realPart, imaginaryPart);
```

```
    }

    return 0;
}
```

OUTPUT

```
Enter coefficients a, b and c of the quadratic equation (ax^2 + bx + c = 0):
1 -3 2
The equation has two real and distinct roots:
Root 1 = 2.00
Root 2 = 1.00

Enter coefficients a, b and c of the quadratic equation (ax^2 + bx + c = 0):
1 -2 1
The equation has two real and equal roots:
Root 1 = 1.00
Root 2 = 1.00

Enter coefficients a, b and c of the quadratic equation (ax^2 + bx + c = 0):
1 2 5
The equation has two complex roots:
Root 1 = -1.00 + 2.00i
Root 2 = -1.00 - 2.00i
```

2. 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 amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.

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

int main()
{
    char userName[50];
    int units, meterCharge = 100;
    float amount = 0, surcharge = 0, totalAmount = 0;

    // Read user name and units consumed
    printf("Enter user name: ");
    scanf("%s", userName);

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

    // Calculate amount based on unit slabs
    if (units <= 200)
    {
        amount = units * 0.80;
    }
    else if (units <= 300)
    {
        amount = (200 * 0.80) + ((units - 200) * 0.90);
    }
    else
    {
        amount = (200 * 0.80) + (100 * 0.90) + ((units - 300) *
1.00);
    }

    // Add minimum meter charge of Rs. 100
    totalAmount = amount + meterCharge;
```

```c
    // Apply surcharge if total amount exceeds Rs. 400
    if (totalAmount > 400)
    {
        surcharge = totalAmount * 0.15;
        totalAmount = totalAmount + surcharge;
    }

    // Print the result
    printf("\nElectricity Bill\n");
    printf("User Name      : %s\n", userName);
    printf("Units Consumed: %d\n", units);
    printf("Amount (Rs)    : %.2f\n", amount);
    printf("Meter Charge   : 100.00\n");

    if (surcharge > 0)
    {
        printf("Surcharge (15%): %.2f\n", surcharge);
    }

    printf("Total Amount   : %.2f\n", totalAmount);

    return 0;
}
```

OUTPUT

Enter user name: Ravi
Enter number of units consumed: 150
Electricity Bill
User Name : Ravi
Units Consumed: 150
Amount (Rs) : 120.00
Meter Charge : 100.00
Total Amount : 220.00

Enter user name: Neha
Enter number of units consumed: 250
Electricity Bill
User Name : Neha
Units Consumed: 250
Amount (Rs) : 170.00
Meter Charge : 100.00
Total Amount : 270.00

```
Enter user name: Divya
Enter number of units consumed: 400
Electricity Bill
User Name     : Divya
Units Consumed: 400
Amount (Rs)   : 405.00
Meter Charge  : 100.00
Surcharge (15%): 75.75
Total Amount  : 580.75
```

3. 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 3 4 3 2 1
  1 2 3 2 1
    1 2 1
      1
```

```c
#include <stdio.h>

int main()
{
    int totalRows, row, col, space;

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

    // Upper half
    for (row = 1; row <= totalRows; row++)
    {
        // Print leading spaces
        for (space = 1; space <= totalRows - row; space++)
        {
            printf("  ");
        }

        // Print increasing numbers
        for (col = 1; col <= row; col++)
        {
            printf("%d ", col);
        }

        // Print decreasing numbers
        for (col = row - 1; col >= 1; col--)
        {
            printf("%d ", col);
        }

        printf("\n");
```

```
    }

    // Lower half
    for (row = totalRows - 1; row >= 1; row--)
    {
        // Print leading spaces
        for (space = 1; space <= totalRows - row; space++)
        {
            printf("  ");
        }

        // Print increasing numbers
        for (col = 1; col <= row; col++)
        {
            printf("%d ", col);
        }

        // Print decreasing numbers
        for (col = row - 1; col >= 1; col--)
        {
            printf("%d ", col);
        }

        printf("\n");
    }

    return 0;
}
```

OUTPUT

```
Enter the number of rows: 5
        1
      1 2 1
    1 2 3 2 1
  1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
  1 2 3 4 3 2 1
    1 2 3 2 1
      1 2 1
        1
```

4. Implement Binary Search on Integers.

```c
#include <stdio.h>

int binarySearch(int arr[], int size, int key)
{
    int low = 0;
    int high = size - 1;

    while (low <= high)
    {
        int mid = (low + high) / 2;

        if (arr[mid] == key)
        {
            return mid;
        }
        else if (arr[mid] < key)
        {
            low = mid + 1;
        }
        else
        {
            high = mid - 1;
        }
    }

    return -1;
}

int main()
{
    int size;
    printf("Enter the number of elements: ");
    scanf("%d", &size);

    int arr[size];

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

```
    int searchKey;
    printf("Enter the number to search: ");
    scanf("%d", &searchKey);

    int resultIndex = binarySearch(arr, size, searchKey);

    if (resultIndex != -1)
    {
        printf("Number found at index %d \n", resultIndex);
    }
    else
    {
        printf("Number not found in the array.\n");
    }

    return 0;
}
```

OUTPUT

Enter the number of elements: 5
Enter 5 sorted integers:
23
56
879
1212
5674
Enter the number to search: 1212
Number found at index 4

5. Implement Matrix multiplication and validate the rules of multiplication.

```c
#include <stdio.h>

int main()
{
    int rowsA, colsA, rowsB, colsB;

    printf("Enter rows and columns of Matrix A: ");
    scanf("%d %d", &rowsA, &colsA);

    printf("Enter rows and columns of Matrix B: ");
    scanf("%d %d", &rowsB, &colsB);

    // Validate multiplication rule
    if (colsA != rowsB)
    {
        printf("Matrix multiplication not possible. Columns of A must
equal rows of B.\n");
        return 0;
    }

    int matrixA[rowsA][colsA];
    int matrixB[rowsB][colsB];
    int resultMatrix[rowsA][colsB];

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

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

```
    }

    // Initialize result matrix to zero
    for (int i = 0; i < rowsA; i++)
    {
        for (int j = 0; j < colsB; j++)
        {
            resultMatrix[i][j] = 0;
        }
    }

    // Multiply matrices
    for (int i = 0; i < rowsA; i++)
    {
        for (int j = 0; j < colsB; j++)
        {
            for (int k = 0; k < colsA; k++)
            {
                resultMatrix[i][j] += matrixA[i][k] * matrixB[k][j];
            }
        }
    }

    // Display Result
    printf("Resultant Matrix (A x B):\n");
    for (int i = 0; i < rowsA; i++)
    {
        for (int j = 0; j < colsB; j++)
        {
            printf("%d\t", resultMatrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

OUTPUT

```
Enter rows and columns of Matrix A: 3 3
Enter rows and columns of Matrix B: 3 3
Enter elements of Matrix A:
1 2 3
4 5 6
7 8 9
Enter elements of Matrix B:
9 8 7
6 5 4
3 2 1
Resultant Matrix (A x B):
30      24      18
84      69      54
138     114     90
```

6. 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.

$\text{Sin } x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots\cdots$ Taylor Expansion of Sin X	$\text{Cos } x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots\cdots$ Taylor Expansion of Cos X

```c
#include <stdio.h>
#include <math.h>
#define PI 3.14  // More accurate value of π

double factorial(double n)
{
    double fact = 1;
    for (int i = 1; i <= n; i++)
    {
        fact *= i;
    }
    return fact;
}

int main()
{
    double x, r, sinValue = 0;
    int neg = 1;

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

    r = (x * PI) / 180.0;
    printf("Converted to radians: %lf\n", r);

    for (int i = 1; i <= 25; i += 2)
    {
        sinValue += (pow(r, i) / factorial(i)) * neg;
        neg *= -1;
    }
```

```
    printf("Using Taylor Series: sin(%.2lf) = %.6lf\n", x, sinValue);
    printf("Using math.h       : sin(%.2lf) = %.6lf\n", x, sin(r));

    return 0;
}
```

OUTPUT

Enter the value of x (in degrees): 30
Converted to radians: 0.523333
Using Taylor Series: sin(30.00) = 0.499770
Using math.h : sin(30.00) = 0.499770

7. Using functions sort the given set of N numbers (Bubble sort).

```c
#include <stdio.h>

// Function to perform bubble sort
void bubbleSort(int arr[], int size)
{
    for (int i = 0; i < size - 1; i++)
    {
        for (int j = 0; j < size - 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;
            }
        }
    }
}

// Function to print the array
void printArray(int arr[], int size)
{
    printf("Sorted array: ");
    for (int i = 0; i < size; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main()
{
    int numbers[100], n;

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

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

```
    }

    bubbleSort(numbers, n);
    printArray(numbers, n);

    return 0;
}
```

OUTPUT

Enter the number of elements: 5
Enter 5 integers:
23
21
5
4
2
Sorted array: 2 4 5 21 23

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

```c
#include <stdio.h>

// Function to find and print the length of a string
void findLength(char str[])
{
    int length = 0;
    while (str[length] != '\0')
    {
        length++;
    }
    printf("Length = %d\n", length);
}

// Function to compare two strings
void compareStrings(char str1[], char str2[])
{
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0')
    {
        if (str1[i] != str2[i])
        {
            printf("Comparison result: Strings are not equal.\n");
            return;
        }
        i++;
    }

    if (str1[i] == '\0' && str2[i] == '\0')
        printf("Comparison result: Strings are equal.\n");
    else
        printf("Comparison result: Strings are not equal.\n");
}

// Function to concatenate and print result
void concatenateStrings(char str1[], char str2[])
{
    int i = 0, j = 0;

    while (str1[i] != '\0')
        i++;
```

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

    str1[i] = '\0';

    printf("Concatenated string: %s\n", str1);
}

int main()
{
    char string1[100], string2[50];

    printf("Enter first string: ");
    scanf("%s", string1);

    printf("Enter second string: ");
    scanf("%s", string2);

    compareStrings(string1, string2);

    printf("Length of the first string: ");
    findLength(string1);

    printf("Length of the second string: ");
    findLength(string2);

    concatenateStrings(string1, string2);

    return 0;
}
```

OUTPUT

Enter first string: bms
Enter second string: it
Comparison result: Strings are not equal.
Length of the first string: Length = 3
Length of the second string: Length = 2
Concatenated string: bmsit

9. 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
#include <stdio.h>

struct Student
{
    char name[50];
    float marks;
};

int main()
{
    int numberOfStudents, i;
    float sum = 0.0, average;

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

    struct Student students[numberOfStudents];

    for (i = 0; i < numberOfStudents; 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;
    }

    average = sum / numberOfStudents;
    printf("\nAverage marks of the class: %f\n", average);

    printf("\nStudents scoring above average:\n");
    for (i = 0; i < numberOfStudents; i++)
    {
        if (students[i].marks > average)
        {
            printf("%s with marks %f\n", students[i].name,
students[i].marks);
        }
    }
```

```
    printf("\nStudents scoring below average:\n");
    for (i = 0; i < numberOfStudents; i++)
    {
        if (students[i].marks < average)
        {
            printf("%s with marks %f\n", students[i].name,
students[i].marks);
        }
    }

    return 0;
}
```

OUTPUT

Enter the number of students: 5
Enter name of student 1: shankar
Enter marks of shankar: 92
Enter name of student 2: guru
Enter marks of guru: 65
Enter name of student 3: shwetha
Enter marks of shwetha: 98
Enter name of student 4: lakshmi
Enter marks of lakshmi: 78
Enter name of student 5: hareesh
Enter marks of hareesh: 42

Average marks of the class: 75.000000

Students scoring above average:
shankar with marks 92.000000
shwetha with marks 98.000000
lakshmi with marks 78.000000

Students scoring below average:
guru with marks 65.000000
hareesh with marks 42.000000

10. 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
#include <stdio.h>
#include <math.h>

int main()
{
    int n, i;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    float arr[n];
    float *ptr = arr;

    printf("Enter %d elements ", n);
    for (i = 0; i < n; i++)
    {
        scanf("%f", ptr + i);
    }

    // Compute sum
    float sum = 0.0f;
    for (i = 0; i < n; i++)
    {
        sum += *(ptr + i);
    }

    // Compute mean
    float mean = sum / n;

    // Compute standard deviation
    float varianceSum = 0.0f;
    for (i = 0; i < n; i++)
    {
        float diff = *(ptr + i) - mean;
        varianceSum = varianceSum + (diff * diff);
    }
    float stdDeviation = sqrt(varianceSum / n);

    printf("\nSum = %f", sum);
    printf("\nMean = %f", mean);
    printf("\nStandard Deviation = %f\n", stdDeviation);
```

```
    return 0;
}
```

OUTPUT

Enter the number of elements: 5
Enter 5 elements 1 2 3 4 5

Sum = 15.000000
Mean = 3.000000
Standard Deviation = 1.414214