WEEK 11

Objective: Explore the Functions, sub-routines, scope and extent of variables, doing some experiments by parameter passing using call by value. Basic methods of numerical integration .

Suggested Experiments/Activities:

Tutorial 11: Functions, call by value, scope and extent,

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

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

i) Write a C function to calculate NCR value.

Aim: To write a C program that defines a function to calculate the nCr (combination) value using the formula:

$$nCr = \frac{n!}{r!(n-r)!}$$

and display the result for the given values of n and r.

Description: nCr is a mathematical formula used to find the number of ways to choose **r items from n items** *without repetition and without order*.

Example: If you have 5 people and want to choose 2, the number of possible groups is **5C2 = 10**.

Algorithm: To Calculate nCr Using Function

1. **Start**
2. **Read** the values of **n** and **r** from the user.
3. **Define a function** factorial(int x)
   - Initialize fact = 1
   - Repeat from i = 1 to x
     - Multiply fact = fact * i
   - Return fact.
4. **Define another function** nCr(int n, int r)
   - If r > n, return **0** (Invalid case).
   - Compute num = factorial(n)
   - Compute den = factorial(r) * factorial(n - r)
   - Compute result = num / den
   - Return result.
5. **Call the function** nCr(n, r) and store the returned value.
6. **Display** the nCr value.
7. **Stop**

Program:

```
#include <stdio.h>
int factorial(int n)
{
   int fact = 1;
   for(int i = 1; i <= n; i++) {
      fact = fact * i;
   }
   return fact;
}

int nCr(int n, int r)
{
   int num = factorial(n);
   int den = factorial(r) * factorial(n - r);
   return num / den;
}
void main()
{
   int n, r;
   printf("Enter n: ");
   scanf("%d", &n);
   printf("Enter r: ");
   scanf("%d", &r);
   if(r > n)
  {
      printf("Invalid! r cannot be greater than n.\n");
   }
   else
  {
      printf("nCr value = %d\n", nCr(n, r));
   }
  }
```

Expected Input:
Enter n: 5
Enter r: 2
Expected output:
nCr value = 10

Expected Input:
Enter n: 2
Enter r: 5
Expected output:
Invalid! r cannot be greater than n.

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

Aim: To write a C program that defines a function to find the length of a string without using built-in functions and display the result.

Algorithm: To Find Length of a String Using Function

1. **Start**
2. **Declare** a character array str to store the string.
3. **Read** the string from the user.
4. **Define a function** stringLength(str)
    - Initialize length = 0
    - Repeat while str[length] is not '\0'
        - Increment length
    - Return length
5. **Call** the function stringLength(str) and store the result.
6. **Display** the length of the string.
7. **Stop**

Program:

```
#include <stdio.h>
int stringLength(char str[])
{
   int length = 0;
   while (str[length] != '\0')
     {
      length++;
     }
   return length;
}
Void main( )
{
   char str[100];
   printf("Enter a string: ");
   gets(str);
   int len = stringLength(str);
   printf("Length of the string = %d\n", len);
}
```

Expected Input:

Enter a string: Rajani

Expected output:

Length of the string = 6

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

Aim: To write a C program that uses a function to find and display the transpose of a matrix**.**

Algorithm: Transpose of a Matrix Using Function

1. **Start**
2. **Declare** a 2D array a to store the original matrix and another 2D array t to store the transpose.
3. **Read** the number of rows m and columns n.
4. **Read** the elements of matrix a[m][n].
5. **Call the function** transpose(a, t, m, n)

Inside the function:

- For each row i from 0 to m−1
  - For each column j from 0 to n−1
    - Set t[j][i] = a[i][j]

6. **Display** the transpose matrix t.
7. **Stop**

Program:

```
#include <stdio.h>
void transpose(int a[10][10], int t[10][10], int m, int n)
{
   for(int i = 0; i < m; i++)
    {
      for(int j = 0; j < n; j++)
       {
          t[j][i] = a[i][j];
       }
    }
}
void main( )
{
   int a[10][10], t[10][10];
   int m, n;
   printf("Enter number of rows: ");
   scanf("%d", &m);
   printf("Enter number of columns: ");
   scanf("%d", &n);
   printf("Enter elements of the matrix:\n");
   for(int i = 0; i < m; i++)
   {
       for(int j = 0; j < n; j++)
       {
          scanf("%d", &a[i][j]);
       }
   }
   transpose(a, t, m, n);
   printf("\nTranspose of the matrix:\n");
   for(int i = 0; i < n; i++) {
      for(int j = 0; j < m; j++)
      {
         printf("%d ", t[i][j]);
      }
      printf("\n");
```

```
}
}
```

Expected Input:

Enter number of rows: 2

Enter number of columns: 2

Enter elements of the matrix:

1 2

3 4

Expected Output:

Transpose of the matrix:

1 3

2 4

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

Aim: To write a C program that uses a function to demonstrate numerical integration of a first-order differential equation using Euler's Method.

The program computes the approximate solution of dy/dx = f(x, y) with given initial conditions.

Algorithm: Euler's Method Using Function

1. **Start**
2. **Define** the differential equation in the form : dy/dx=f(x,y)
3. **Read** initial values:
    1. x0 → initial x
    2. y0 → initial y
    3. h → step size
    4. n → number of steps
4. **Call the Euler function** euler(x0, y0, h, n)

    Repeat n times:

    1. Compute slope: k=f(x,y)
    2. Update y: y=y+h·k
    3. Update x: x=x+h
    4. Print the value of x and y at each step
5. **Display** the final approximate solution.
6. **Stop**

Program:

```
#include <stdio.h>
float f(float x, float y)
{
```

```
    return x + y;
}
void euler(float x0, float y0, float h, int n)
{
    float x = x0;
    float y = y0;
    printf("\nEuler's Method Steps:\n");
    printf("x\t\t y\n");
    for(int i = 0; i < n; i++)
     {
        float k = f(x, y);
        y = y + h * k;
        x = x + h;
        printf("%.4f\t %.4f\n", x, y);
    }
    printf("\nFinal Approximate Value: y(%.2f) = %.4f\n", x, y);
}
void main( )
{
    float x0, y0, h;
    int n;
    printf("Enter initial value of x (x0): ");
    scanf("%f", &x0);
    printf("Enter initial value of y (y0): ");
    scanf("%f", &y0);
    printf("Enter step size (h): ");
    scanf("%f", &h);
    printf("Enter number of steps (n): ");
    scanf("%d", &n);
    euler(x0, y0, h, n);
}
```

Expected Input and Output:

```
Enter initial value of x (x0): 10
Enter initial value of y (y0): 2
Enter step size (h): 2
Enter number of steps (n): 4
Euler's Method Steps:
x               y
12.0000         26.0000
14.0000         102.0000
16.0000         334.0000
18.0000         1034.0000
Final Approximate Value: y(18.00) = 1034.0000
```