# ASSIGNMENT

**Typedef-**

```c
#include<stdio.h>
typedef int my_int;
int main(){
    my_int a=10; //alias name my_int has been used for declaring the variable
    printf("a = %d",a);
}
```

Output
a=10

**Implementing typedef along with structures-**

```c
#include<stdio.h>
typedef struct date{
    int day;
    int month;
    int year;
}dt;

int main(){
    dt var1={26,11,2024};
    printf("sizeof var1 = %d\n",sizeof(var1));

    printf("Today's Date = %d-%d-%d",var1.day,var1.month,var1.year);
    return 0;
}
```

Output-

```
sizeof var1 = 12
Today's Date = 26-11-2024
```

**Using typedef with pointers-**

```c
#include<stdio.h>
typedef int* intPtr;
int main(){
    int a=20;
    intPtr ptr1=&a;
    printf("001a = %d\n",*ptr1);
    *ptr1=30;
    printf("001a = %d\n",*ptr1);
    return 0;
}
```

Output-

```
001a = 20
001a = 30
```

**Using typedef for arrays-**

```c
#include<stdio.h>

typedef int arr[4];
int main(){
    arr t = {1,2,3,4};
    for(int i=0;i<4;i++){
        printf("%d ",t[i]);
    }
    return 0;
}
```

Output-

```
1 2 3 4
```

Write a program that defines a custom data type Complex using typedef to represent a complex number with real and imaginary parts. Implement functions to:

- Add two complex numbers.
- Multiply two complex numbers.
- Display a complex number in the format "a + bi".

**Input Example**

Enter first complex number (real and imaginary): 3 4

Enter second complex number (real and imaginary): 1 2

**Output Example**

Sum: 4 + 6i

Product: -5 + 10i

```c
#include <stdio.h>

typedef struct {
    float real;
    float imaginary;
} complex;
complex addComplex(complex c1, complex c2);
complex multiplyComplex(complex c1, complex c2);
void displayComplex(complex c);
int main() {
    complex c1, c2, sum, product;
    printf("Enter Real and Imaginary Part of a Complex Number: ");
    scanf("%f %f", &c1.real, &c1.imaginary);
    printf("Enter another Real and Imaginary Part of a Complex Number: ");
    scanf("%f %f", &c2.real, &c2.imaginary);
    sum = addComplex(c1, c2);
    product = multiplyComplex(c1, c2);
    printf("Sum: ");
    displayComplex(sum);
    printf("\n");
    printf("Product: ");
    displayComplex(product);
    printf("\n");
    return 0;
}
complex addComplex(complex c1, complex c2) {
    complex result;
    result.real = c1.real + c2.real;
    result.imaginary = c1.imaginary + c2.imaginary;
    return result;
}
complex multiplyComplex(complex c1, complex c2) {
    complex result;
    result.real = c1.real * c2.real - c1.imaginary * c2.imaginary;
    result.imaginary = c1.real * c2.imaginary + c1.imaginary * c2.real;
    return result;
}
void displayComplex(complex c) {
```

```
    printf("%.2f + %.2fi", c.real, c.imaginary);
}
```

Output:
Enter Real and Imaginary Part of a Complex Number: 3 4
Enter another Real and Imaginary Part of a Complex Number: 1 2
Sum: 4.00 + 6.00i
Product: -5.00 + 10.00i

## Typedef for Structures

**Problem Statement:**
Define a custom data type Rectangle using typedef to represent a rectangle with width and height as float values. Write functions to:

- Compute the area of a rectangle.
- Compute the perimeter of a rectangle.

**Input Example:**

Enter width and height of the rectangle: 5 10

**Output Example**:

Area: 50.00

Perimeter: 30.00

```c
#include <stdio.h>
typedef struct {
    float width;
    float height;
} Rectangle;
float computeArea(Rectangle rect);
float computePerimeter(Rectangle rect);
int main() {
    Rectangle rect;
    float area, perimeter;
    printf("Enter width and height of the rectangle: ");
    scanf("%f %f", &rect.width, &rect.height);
    area = computeArea(rect);
    perimeter = computePerimeter(rect);
    printf("Area: %.2f\n", area);
    printf("Perimeter: %.2f\n", perimeter);
    return 0;
}
float computeArea(Rectangle rect) {
    return rect.width * rect.height;
}
float computePerimeter(Rectangle rect) {
    return 2 * (rect.width + rect.height);
}
```

Output-
Enter width and height of the rectangle: 2
5
Area: 10.00
Perimeter: 14.00

**Function Pointers-**

```c
#include<stdio.h>
void display(int);
int main(){
    //Declaring a pointer to the function display()
    // void (*func_ptr)(int)=&display;
    void(*func_ptr)(int);
    func_ptr=&display;//initializing the pointer with the address of function display()
    (*func_ptr)(20);//calling the function as well passing the parameter using function
pointers

    return 0;
}
void display(int a){
    printf("a = %d",a);
}
```

Output:

```
a = 20
```

**Array of Function Pointers-**

```c
#include <stdio.h>

void add(int , int);
void sub(int , int);
void mul(int , int);

int main(){
    void (*fun_ptr_arr[])(int, int) = {add, sub, mul};
    int a = 10, b = 20;
    (*fun_ptr_arr[0])(a,b);
    (*fun_ptr_arr[1])(a,b);
    (*fun_ptr_arr[2])(a,b);
    return 0;
}

void add(int a, int b){
    int sum = a + b;
    printf("sum = %d \n",sum);
}
void sub(int a, int b){
    int sub = a - b;
    printf("sub = %d \n",sub);
}
void mul(int a, int b){
    int mul = a * b;
    printf("mul = %d \n",mul);
}
```

Output:

```
sum = 30
sub = -10
mul = 200
```

**Simple Calculator Using Function Pointers**

**Problem Statement:**

Write a C program to implement a simple calculator. Use function pointers to dynamically call functions for addition, subtraction, multiplication, and division based on user input.

**Input Example:**

Enter two numbers: 10 5

Choose operation (+, -, *, /): *

**Output Example:**

Result: 50

```c
#include<stdio.h>

void add(int ,int);
void sub(int ,int);
void mul(int ,int);
void div(int ,int);
int main(){
    void (*fun_ptr_arr[])(int,int)={add,sub,mul,div};
    int a,b;
    char operation;
    printf("Enter First Number");
    scanf("%d",&a);
    printf("Enter Second Number");
    scanf("%d",&b);
    getchar();
    printf("Choose Operation(+ - * /): ");
    scanf("%c",&operation);
    switch(operation){
        case '+':
            (*fun_ptr_arr[0])(a,b);
            break;
        case '-':
            (*fun_ptr_arr[1])(a,b);
            break;
        case '*':
            (*fun_ptr_arr[2])(a,b);
            break;
        case '/':
            (*fun_ptr_arr[3])(a,b);
            break;
        default:
            printf("Choose a Valid Operation");
    }
    return 0;
}
void add(int a,int b){
    int sum=a+b;
    printf("Result = %d\n",sum);
}
void sub(int a,int b){
    int difference=a-b;
    printf("Result = %d\n",difference);
}
void mul(int a,int b){
    int product=a*b;
    printf("Result = %d\n",product);
}
void div(int a,int b){
    if(b==0){
        printf("Division by Zero not Possible");
    }
    else{
```

```
        int division=a/b;
        printf("Result = %.2f",division);
    }
}
```

Output:
Enter First Number2
Enter Second Number5
Choose Operation(+ - * /): -
Result = -3

**Array Operations Using Function Pointers**

**Problem Statement:**
Write a C program that applies different operations to an array of integers using function pointers. Implement operations like finding the maximum, minimum, and sum of elements.

**Input Example:**

Enter size of array: 4

Enter elements: 10 20 30 40

Choose operation (1 for Max, 2 for Min, 3 for Sum): 3

**Output Example:**

Result: 100

```
#include <stdio.h>

int findMax(int arr[], int size);
int findMin(int arr[], int size);
int findSum(int arr[], int size);
int main() {
    int size, i, choice;
    int result;
    printf("Enter size of array: ");
    scanf("%d", &size);
    int arr[size];
    printf("Enter %d elements for array: ",size);
    for (i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    int (*operations[])(int[], int) = {findMax, findMin, findSum};
    printf("Choose operation (1 for Max, 2 for Min, 3 for Sum): ");
    scanf("%d", &choice);
    if (choice < 1 || choice > 3) {
        printf("Invalid operation!\n");
        return 1;
    }
    result = (*operations[choice - 1])(arr, size);
    printf("Result: %d\n", result);
    return 0;
}
int findMax(int arr[], int size) {
    int max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
```

```
            max = arr[i];
        }
    }
    return max;
}
int findMin(int arr[], int size) {
    int min = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }
    return min;
}
int findSum(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}
```

Output:
Enter size of array: 4
Enter 4 elements for array: 23
54
11
9
Choose operation (1 for Max, 2 for Min, 3 for Sum): 2
Result: 9

**Event System Using Function Pointers**

**Problem Statement:**
Write a C program to simulate a simple event system. Define three events: onStart, onProcess, and onEnd. Use function pointers to call appropriate event handlers dynamically based on user selection.

**Input Example:**

Choose event (1 for onStart, 2 for onProcess, 3 for onEnd): 1

**Output Example:**

Event: onStart
Starting the process…

```
#include<stdio.h>

void onStart();
void onProcess();
void onEnd();
int main(){
    void (*Event_System[])(void)={onStart,onProcess,onEnd};
    int choice;
    printf("Enter your Choice : \n 1=>onStart \n 2=>onProcess \n 3=>onEnd \t");
    scanf("%d",&choice);
    switch(choice){
        case 1:
            (*Event_System[0])();
```

```
            break;
        case 2:
            (*Event_System[1])();
            break;
        case 3:
            (*Event_System[2])();
            break;
        default:
            printf("InValid Choice");
    }
    return 0;
}
void onStart(){
    printf("onStart\n Starting the Process...");
}
void onProcess(){
    printf("onProcess\n Processing...");
}
void onEnd(){
    printf("onEnd\n Ending the Process...");
}
```

Output:
Enter your Choice :
1=>onStart
2=>onProcess
3=>onEnd 2
onProcess
Processing...

**Matrix Operations with Function Pointers**

**Problem Statement:**
Write a C program to perform matrix operations using function pointers. Implement functions to add, subtract, and multiply matrices. Pass the function pointer to a wrapper function to perform the desired operation.

**Input Example:**

Enter matrix size (rows and columns): 2 2
Enter first matrix:
1 2
3 4
Enter second matrix:
5 6
7 8
Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): 1

**Output Example:**

Result:
6 8
10 12

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

```c
void add(int **mat1, int **mat2, int r, int c);
void sub(int **mat1, int **mat2, int r, int c);
void mul(int **mat1, int **mat2, int r1, int c1, int c2);
int main() {
    void (*fun_ptr_arr[3])(int **mat1, int **mat2, int, int) = {add, sub, NULL};
    int **mat1, **mat2, **result;
    int r, c, op;
    printf("Enter matrix size (rows and columns): ");
    scanf("%d %d", &r, &c);

    mat1 = (int **)malloc(r * sizeof(int *));
    mat2 = (int **)malloc(r * sizeof(int *));
    result = (int **)malloc(r * sizeof(int *));
    for (int i = 0; i < r; i++) {
        mat1[i] = (int *)malloc(c * sizeof(int));
        mat2[i] = (int *)malloc(c * sizeof(int));
        result[i] = (int *)malloc(c * sizeof(int));
    }

    printf("Enter first matrix:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            scanf("%d", &mat1[i][j]);
        }
    }
    printf("Enter second matrix:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            scanf("%d", &mat2[i][j]);
        }
    }
    printf("Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): ");
    scanf("%d", &op);
    switch (op) {
        case 1:
            (*fun_ptr_arr[0])(mat1, mat2, r, c);
            break;
        case 2:
            (*fun_ptr_arr[1])(mat1, mat2, r, c);
            break;
        case 3:
            mul(mat1, mat2, r, c, c);
            break;
        default:
            printf("Invalid option!!\n");
            break;
    }

    for (int i = 0; i < r; i++) {
        free(mat1[i]);
        free(mat2[i]);
        free(result[i]);
    }
    free(mat1);
    free(mat2);
    free(result);
    return 0;
}
void add(int **mat1, int **mat2, int r, int c) {
    printf("Result:\n");
    for (int i = 0; i < r; i++) {
```

```
        for (int j = 0; j < c; j++) {
            printf("%d ", mat1[i][j] + mat2[i][j]);
        }
        printf("\n");
    }
}
void sub(int **mat1, int **mat2, int r, int c) {
    printf("Result:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            printf("%d ", mat1[i][j] - mat2[i][j]);
        }
        printf("\n");
    }
}
void mul(int **mat1, int **mat2, int r1, int c1, int c2) {
    int **result = (int **)malloc(r1 * sizeof(int *));
    for (int i = 0; i < r1; i++) {
        result[i] = (int *)malloc(c2 * sizeof(int));
    }
    printf("Result:\n");
    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] += mat1[i][k] * mat2[k][j];
            }
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
    for (int i = 0; i < r1; i++) {
        free(result[i]);
    }
    free(result);
}
```

Output

Enter matrix size (rows and columns): 2
3
Enter first matrix:
4
5
6
7
7
8
Enter second matrix:
3
5
6
7
7
6
Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): 1
Result:
7 10 12
14 14 14

**Problem Statement: Vehicle Management System**

Write a C program to manage information about various vehicles. The program should demonstrate the following:

1. **Structures**: Use structures to store common attributes of a vehicle, such as vehicle type, manufacturer name, and model year.
2. **Unions**: Use a union to represent type-specific attributes, such as:
    1. Car: Number of doors and seating capacity.
    2. Bike: Engine capacity and type (e.g., sports, cruiser).
    3. Truck: Load capacity and number of axles.
3. **Typedefs**: Define meaningful aliases for complex data types using typedef (e.g., for the structure and union types).
4. **Bitfields**: Use bitfields to store flags for vehicle features like **airbags**, **ABS**, and **sunroof**.
5. **Function Pointers**: Use a function pointer to dynamically select a function to display specific information about a vehicle based on its type.

## Requirements

1. Create a structure Vehicle that includes:
    1. A char array for the manufacturer name.
    2. An integer for the model year.
    3. A union VehicleDetails for type-specific attributes.
    4. A bitfield to store vehicle features (e.g., airbags, ABS, sunroof).
    5. A function pointer to display type-specific details.
2. Write functions to:
    1. Input vehicle data, including type-specific details and features.
    2. Display all the details of a vehicle, including the type-specific attributes.
    3. Set the function pointer based on the vehicle type.
3. Provide a menu-driven interface to:
    1. Add a vehicle.
    2. Display vehicle details.
    3. Exit the program.

# Example Input/Output

### Input:

1. Add Vehicle
2. Display Vehicle Details

3. Exit
Enter your choice: 1

Enter vehicle type (1: Car, 2: Bike, 3: Truck): 1
Enter manufacturer name: Toyota
Enter model year: 2021
Enter number of doors: 4
Enter seating capacity: 5
Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): 1 1 0

1. Add Vehicle
2. Display Vehicle Details
3. Exit
Enter your choice: 2

Output:

Manufacturer: Toyota

Model Year: 2021

Type: Car

Number of Doors: 4

Seating Capacity: 5

Features: Airbags: Yes, ABS: Yes, Sunroof: No

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

typedef union {
    struct {
        int numberOfDoors;
        int seatingCapacity;
    } car;
    struct {
        int engineCapacity;
        char type[10];
    } bike;
    struct {
        int loadCapacity;
        int numberOfAxles;
    } truck;
} VehicleDetails;
typedef struct {
    unsigned int airbags : 1;
    unsigned int abs : 1;
    unsigned int sunroof : 1;
} VehicleFeatures;
typedef struct Vehicle {
    char manufacturer[50];
    int modelYear;
    int vehicleType;
```

```c
    VehicleDetails details;
    VehicleFeatures features;
    void (*displayDetails)(struct Vehicle *v);
} Vehicle;
void inputVehicle(Vehicle *v);
void displayCarDetails(Vehicle *v);
void displayBikeDetails(Vehicle *v);
void displayTruckDetails(Vehicle *v);
void setDisplayFunction(Vehicle *v);
void displayVehicle(Vehicle *v);
int main() {
    Vehicle vehicles[10];
    int vehicleCount = 0;
    int choice;
    while (1) {
        printf("\n1. Add Vehicle\n2. Display Vehicle Details\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                if (vehicleCount < 10) {
                    inputVehicle(&vehicles[vehicleCount]);
                    setDisplayFunction(&vehicles[vehicleCount]);
                    vehicleCount++;
                } else {
                    printf("Vehicle storage is full!\n");
                }
                break;
            case 2:
                for (int i = 0; i < vehicleCount; i++) {
                    printf("\nVehicle %d Details:\n", i + 1);
                    displayVehicle(&vehicles[i]);
                }
                break;
            case 3:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
    return 0;
}
void inputVehicle(Vehicle *v) {
    printf("Enter manufacturer name: ");
    scanf("%s", v->manufacturer);
    printf("Enter model year: ");
    scanf("%d", &v->modelYear);
    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &v->vehicleType);
    switch (v->vehicleType) {
        case 1:
            printf("Enter number of doors: ");
            scanf("%d", &v->details.car.numberOfDoors);
            printf("Enter seating capacity: ");
            scanf("%d", &v->details.car.seatingCapacity);
            break;
        case 2:
            printf("Enter engine capacity: ");
            scanf("%d", &v->details.bike.engineCapacity);
            printf("Enter type (e.g., sports, cruiser): ");
```

```
            scanf("%s", v->details.bike.type);
            break;
        case 3:
            printf("Enter load capacity: ");
            scanf("%d", &v->details.truck.loadCapacity);
            printf("Enter number of axles: ");
            scanf("%d", &v->details.truck.numberOfAxles);
            break;
        default:
            printf("Invalid vehicle type!\n");
            return;
    }
    unsigned int airbags, abs, sunroof;
    printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
    scanf("%u %u %u", &airbags, &abs, &sunroof);
    v->features.airbags = airbags;
    v->features.abs = abs;
    v->features.sunroof = sunroof;
}
void setDisplayFunction(Vehicle *v) {
    switch (v->vehicleType) {
        case 1:
            v->displayDetails = displayCarDetails;
            break;
        case 2:
            v->displayDetails = displayBikeDetails;
            break;
        case 3:
            v->displayDetails = displayTruckDetails;
            break;
        default:
            v->displayDetails = NULL;
            break;
    }
}
void displayCarDetails(Vehicle *v) {
    printf("Type: Car\n");
    printf("Number of Doors: %d\n", v->details.car.numberOfDoors);
    printf("Seating Capacity: %d\n", v->details.car.seatingCapacity);
}
void displayBikeDetails(Vehicle *v) {
    printf("Type: Bike\n");
    printf("Engine Capacity: %d CC\n", v->details.bike.engineCapacity);
    printf("Type: %s\n", v->details.bike.type);
}
void displayTruckDetails(Vehicle *v) {
    printf("Type: Truck\n");
    printf("Load Capacity: %d Tons\n", v->details.truck.loadCapacity);
    printf("Number of Axles: %d\n", v->details.truck.numberOfAxles);
}
void displayVehicle(Vehicle *v) {
    printf("Manufacturer: %s\n", v->manufacturer);
    printf("Model Year: %d\n", v->modelYear);
    if (v->displayDetails != NULL) {
        v->displayDetails(v);
    }
    printf("Features:\n");
    printf("  Airbags: %s\n", v->features.airbags ? "Yes" : "No");
    printf("  ABS: %s\n", v->features.abs ? "Yes" : "No");
    printf("  Sunroof: %s\n", v->features.sunroof ? "Yes" : "No");
}
```

Output
1. Add Vehicle
2. Display Vehicle Details
3. Exit
Enter your choice: 1
Enter manufacturer name: ddd
Enter model year: 45
Enter vehicle type (1: Car, 2: Bike, 3: Truck): 1
Enter number of doors: 4
Enter seating capacity: 6
Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): 1 1 0

1. Add Vehicle
2. Display Vehicle Details
3. Exit
Enter your choice: 2

Vehicle 1 Details:
Manufacturer: ddd
Model Year: 45
Type: Car
Number of Doors: 4
Seating Capacity: 6
Features:
 Airbags: Yes
 ABS: Yes
 Sunroof: No

1. Add Vehicle
2. Display Vehicle Details
3. Exit
Enter your choice: 3
Exiting…

**RECURSION**

1. Write a program to find the factorial

```c
#include<stdio.h>

int factorial(int);
int main(){
    int n;
    printf("Enter a number");
    scanf("%d",&n);
    int fact=factorial(n);
    printf("Factorial = %d",fact);
    return 0;
}
int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
```

Output
Enter a number5
Factorial = 120

2. WAP to find the sum of digits of a number using recursion.

```c
#include <stdio.h>

int sum_of_digits(int n);
int main() {
    int num, sum;
    printf("Enter a number: ");
    scanf("%d", &num);
    sum = sum_of_digits(num);
    printf("Sum of digits of %d is %d\n", num, sum);
    return 0;
}
int sum_of_digits(int n) {
    if (n == 0) {
        return 0;
    } else {
        return (n % 10) + sum_of_digits(n / 10);
    }
}
```

Output:
Enter a number: 520
Sum of digits of 520 is 7

3. With Recursion Findout the maximum number in a given array

```c
#include <stdio.h>

int find_max(int arr[], int n);
int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

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

    int max = find_max(arr, n);
    printf("The maximum number in the array is %d\n", max);

    return 0;
}
int find_max(int arr[], int n) {
    if (n == 1) {
        return arr[0];
    }

    int max_in_rest = find_max(arr, n - 1);

    if (arr[n - 1] > max_in_rest) {
        return arr[n - 1];
    } else {
        return max_in_rest;
    }
}
```

Output
Enter the size of the array: 5
Enter the elements of the array: 23
90
45
80

97
The maximum number in the array is 97

4. With recurion calculate the power of a given number

```c
#include <stdio.h>

int power(int base, int exponent);
int main() {
    int base, exponent, result;
    printf("Enter the base number: ");
    scanf("%d", &base);
    printf("Enter the exponent: ");
    scanf("%d", &exponent);
    result = power(base, exponent);
    printf("%d raised to the power of %d is %d\n", base, exponent, result);
    return 0;
}
int power(int base, int exponent) {
    if (exponent == 0) {
        return 1;
    } else {
        return base * power(base, exponent - 1);
    }
}
```

Output:
Enter the base number: 2
Enter the exponent: 5
2 raised to the power of 5 is 32

5. With Recursion calculate the length of a string.

```c
#include <stdio.h>

int string_length(const char *str);
int main() {
    char str[50];

    printf("Enter a string: ");
    scanf("%s", str);
    int length = string_length(str);
    printf("Length of the string is %d\n", length);
    return 0;
}
int string_length(const char *str) {
    if (str[0] == '\0') {
        return 0;
    } else {
        return 1 + string_length(str + 1);
    }
}
```

Output:
Enter a string: soumya
Length of the string is 6

6. With recursion revrsal of a string

```c
#include <stdio.h>

void reverse_string(char *str, int start, int end);
int main() {
    char str[100];

    printf("Enter a string: ");
```

```
    scanf("%s", str);
    int length = 0;
    while (str[length] != '\0') {
        length++;
    }

    reverse_string(str, 0, length - 1);
    printf("Reversed string is: %s\n", str);
    return 0;
}
void reverse_string(char *str, int start, int end) {
    if (start >= end) {
        return;
    } else {
        char temp = str[start];
        str[start] = str[end];
        str[end] = temp;
        reverse_string(str, start + 1, end - 1);
    }
}
```

Output
Enter a string: soumya
Reversed string is: aymuos