Problem Statement:
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

```
#include <stdio.h>

// Define a custom data type Rectangle
typedef struct {
    float width;
    float height;
} Rectangle;

// Function to compute the area of a rectangle
float area(Rectangle r) {
    return r.width * r.height;
}

// Function to compute the perimeter of a rectangle
float perimeter(Rectangle r) {
    return 2 * (r.width + r.height);
}

int main() {
    Rectangle r;
    float rect_area, rect_perimeter;
```

```
    // Input width and height of the rectangle
    printf("Enter width and height of the rectangle: ");
    scanf("%f %f", &r.width, &r.height);

    // Compute area and perimeter
    rect_area = area(r);
    rect_perimeter = perimeter(r);

    // Output results
    printf("Area: %.2f\n", rect_area);
    printf("Perimeter: %.2f\n", rect_perimeter);

    return 0;
}
```

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 real;
    float imag;
} Complex;

Complex add_complex(Complex c1, Complex c2)
{
    Complex result;
```

```c
    result.real = c1.real + c2.real;
    result.imag = c1.imag + c2.imag;
    return result;
}
Complex multiply_complex(Complex c1, Complex c2)
{
    Complex result;
    result.real = c1.real * c2.real - c1.imag * c2.imag;
    result.imag = c1.real * c2.imag + c1.imag * c2.real;
    return result;
}

void display_complex(Complex c) {
    printf("%.2f + %.2fi\n", c.real, c.imag);
}
int main()
{
    Complex c1, c2, sum, product;
    printf("Enter first complex number (real and imaginary): ");
    scanf("%f %f", &c1.real, &c1.imag);
    printf("Enter second complex number (real and imaginary): ");
    scanf("%f %f", &c2.real, &c2.imag);
    sum = add_complex(c1, c2);
    product = multiply_complex(c1, c2);
    printf("Sum: ");
    display_complex(sum);
    printf("Product: ");
    display_complex(product);
    return 0;
}
```

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()
{
    char ch;
    int num1,num2;
    void (*ptr[])(int,int)={add,sub,mul,div};
    printf("Enter 2 numbers \n");
    scanf("%d %d",&num1,&num2);
    getchar();
    printf("enter the operator \n");
    scanf("%c",&ch);
    switch(ch)
    {
        case '+':
            (*ptr[0])(num1,num2);
            break;
        case '-':
            (*ptr[1])(num1,num2);
            break;
        case '*':
            (*ptr[2])(num1,num2);
            break;
        case '/':
            (*ptr[3])(num1,num2);
            break;
        default :
            printf("wrong operator \n");
    }
    return 0;
}
void add(int a, int b)
{
    printf("sum= %d \n",a+b);
}
void sub(int a, int b)
{
    printf("difference = %d \n",a-b);
```

```
}
void mul(int a, int b)
{
    printf("product = %d \n",a*b);
}
void div(int a, int b)
{
    if (b==0)
        printf("division not possible \n");
    else
        printf("division = %d \n",a/b);
}
```

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>
void max(int [], int);
void min(int [], int);
void sum(int [], int);
int main()
{
    int n,choice;
```

```
    void (*ptr[])(int [], int)={max,min,sum};
    printf("enter the number of elements of the array \n");
    scanf("%d",&n);
    int arr[n];
    printf("enter the elements of the array \n");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("enter the choice \n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
            (*ptr[0])(arr, n);
            break;
        case 2:
            (*ptr[1])(arr,n);
            break;
        case 3:
            (*ptr[2])(arr,n);
            break;
        default:
            printf("wrong choice \n");
            break;
    }
    return 0;
}
void max(int arr1[],int size)
{
    int max=0;
    for(int i=0;i<size;i++)
    {
        if(max<arr1[i])
            max=arr1[i];
    }
    printf("the largest element is %d",max);
}
void min(int arr1[],int size)
{
    int min=arr1[0];
    for(int i=0;i<size;i++)
    {
        if(min>arr1[i])
            min=arr1[i];
```

```
    }
    printf("the smallest element is %d",min);
}
void sum(int arr1[],int size)
{
    int sum=0;
    for(int i=0;i<size;i++)
    {
        sum=sum+arr1[i];
    }
    printf("the sum of elements is %d",sum);
}
```

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);
void onprocess(void);
void onend(void);
int main()
{
    int choice;
    void (*ptr[])(void)={onstart,onprocess,onend};
    printf("enter the choice \n");
    scanf("%d",&choice);
    switch(choice)
```

```
    {
        case 1:
            (*ptr[0])();
            break;
        case 2:
            (*ptr[1])();
            break;
        case 3:
            (*ptr[2])();
            break;
        default:
            printf("wrong choice \n");
            break;
    }
    return 0;
}
void onstart()
{
    printf("event: OnStart \nstarting the process \n");
}
void onprocess()
{
    printf("event :OnProcess \nprocess ongoing\n");
}
void onend()
{
    printf("event: OnEnd \nending the process\n");
}
```

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
```

```c
#include <stdio.h>
void add(int[][10],int[][10],int row,int col);
void sub(int[][10],int[][10],int row, int col);
void mul(int[][10],int[][10],int row,int col);
int main()
{
    int r,c,choice;
    void (*ptr[])(int[][10],int[][10],int,int) = {add,sub,mul};
    printf("Enter the matrix size \n");
    scanf("%d %d", &r, &c);
    int arr1[10][10], arr2[10][10];
    printf("Enter the elements of the 1st matrix\n");
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            scanf("%d", &arr1[i][j]);
        }
    }
    printf("Enter the elements of the 2nd matrix\n");
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            scanf("%d", &arr2[i][j]);
        }
    }
    printf("Enter your choice (1: Add, 2: Subtract, 3: Multiply): ");
    scanf("%d", &choice);
    switch (choice)
    {
        case 1:
            ptr[0](arr1, arr2, r, c);
            break;
        case 2:
            ptr[1](arr1, arr2, r, c);
```

```
            break;
        case 3:
            ptr[2](arr1, arr2, r, c);
            break;
        default:
            printf("Wrong choice\n");
    }
    return 0;
}
void add(int ar1[][10], int ar2[][10], int row, int col)
{
    int ar3[10][10];
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            ar3[i][j] = ar1[i][j] + ar2[i][j];
        }
    }
    printf("added matrix \n");
    for (int i = 0;i<row;i++)
    {
        for (int j=0;j<col;j++)
        {
            printf("%d ",ar3[i][j]);
        }
        printf("\n");
    }
}
void sub(int ar1[][10],int ar2[][10],int row,int col)
{
    int ar3[10][10];
    for (int i = 0;i<row;i++)
    {
        for (int j=0;j<col;j++)
        {
            ar3[i][j]=ar1[i][j]-ar2[i][j];
        }
    }
    printf("subtracted matrix :\n");
    for (int i=0;i<row;i++)
    {
        for (int j=0;j<col;j++)
        {
            printf("%d ",ar3[i][j]);
```

```
        }
        printf("\n");
    }
}
void mul(int ar1[][10],int ar2[][10], int row, int col)
{
    int ar3[10][10] = {0};
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            for (int k = 0; k < col; k++)
            {
                ar3[i][j] += ar1[i][k] * ar2[k][j];
            }
        }
    }
    printf("multiplied matrix \n");
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            printf("%d ", ar3[i][j]);
        }
        printf("\n");
    }
}
```

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:
    - Car: Number of doors and seating capacity.
    - Bike: Engine capacity and type (e.g., sports, cruiser).
    - 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:
    - A char array for the manufacturer name.
    - An integer for the model year.
    - A union VehicleDetails for type-specific attributes.
    - A bitfield to store vehicle features (e.g., airbags, ABS, sunroof).
    - A function pointer to display type-specific details.
2. Write functions to:
    - Input vehicle data, including type-specific details and features.
    - Display all the details of a vehicle, including the type-specific attributes.
    - Set the function pointer based on the vehicle type.
3. Provide a menu-driven interface to:
    - Add a vehicle.
    - Display vehicle details.
    - 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

WAP to find out the factorial of a number using recursion.

```c
#include <stdio.h>
int fact(int n);
int main()
{
    int num, res;
    printf("Enter a number \n");
    scanf("%d", &num);
    if (num<0)
    {
        printf("Factorial is not defined for negative numbers.\n");
    }
    else
    {
        res=fact(num);
        printf("Factorial of %d = %d\n",num,res);
    }
    return 0;
}
int fact(int n)
{
    if (n == 0)
        return 1;
    return n * fact(n - 1);
}
```

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

```c
#include <stdio.h>
int sum(int n);
int main()
{
    int num,ans;
    printf("Enter a number \n");
    scanf("%d", &num);
    ans=sum(num);
    printf("Sum of digits of= %d\n",ans);
    return 0;
}
int sum(int n)
{
    int res;
    if (n == 0)
        return 0;
        res=(n%10)+sum(n/10);
    return res;
}
```

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

```c
#include <stdio.h>
int sum(int n);
int main()
{
    int num,ans;
    printf("Enter a number \n");
    scanf("%d", &num);
    ans=sum(num);
    printf("Sum of digits of= %d\n",ans);
    return 0;
}
int sum(int n)
{
    int res;
    if (n == 0)
        return 0;
```

```
        res=(n%10)+sum(n/10);
    return res;
}
```

With Recursion Find out the maximum number in a given array

```
#include <stdio.h>
int find_max(int arr[], int n)
{
    if (n == 1)
    {
        return arr[0];
    }
    int max_of_rest = find_max(arr, n - 1);
    return (arr[n - 1] > max_of_rest) ? arr[n - 1] : max_of_rest;
}
int main()
{
    int n;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    int max_val = find_max(arr, n);
    printf("The largest number in the array is: %d\n", max_val);
    return 0;
}
```

With Recursion calculate the length of a string.

```
#include <stdio.h>
int len(char str[], int index);
int main()
{
```

```
    int length;
    char str[50];
    printf("Enter a string: ");
    gets(str);
    length = len(str, 0);
    printf("The length of the string is: %d\n", length);

    return 0;
}
int len(char str[], int index)
{
    int res;
    if (str[index] == '\0')
        return 0;
    res=1 + len(str,index + 1);
    return res;
}
```

With recursion reversal of a string

With recurion calculate the power of a given number

```
#include <stdio.h>
int power(int , int );
int main()
{
    int base, exp, res;
    printf("Enter the base\n");
    scanf("%d", &base);
    printf("Enter the exponent\n");
    scanf("%d", &exp);
        res = power(base,exp);
        printf("%d raised to the power %d is: %d\n", base, exp, res);
    return 0;
}
int power(int base, int exp)
{
    if (exp == 0)
        return 1;
    return base * power(base,exp-1);
}
```