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

```c
#include <stdio.h>

typedef float Complex[2];

void add(Complex c1, Complex c2, Complex s);

void mul(Complex c1, Complex c2, Complex p);

void print(Complex c);

int main() {

    Complex c1, c2, sum, product;




    printf("Enter first complex number (real and imaginary): ");

    scanf("%f %f", &c1[0], &c1[1]);



    printf("Enter second complex number (real and imaginary): ");

    scanf("%f %f", &c2[0], &c2[1]);
```

```
    add(c1, c2, sum);
    mul(c1, c2, product);

    printf("Sum: ");
    print(sum);
    printf("Product: ");
    print(product);

    return 0;
}
void add(Complex c1, Complex c2, Complex s) {
    s[0] = c1[0] + c2[0];
    s[1] = c1[1] + c2[1];
}


void mul(Complex c1, Complex c2, Complex p) {
    p[0] = c1[0] * c2[0] - c1[1] * c2[1];
    p[1] = c1[0] * c2[1] + c1[1] * c2[0];
}

void print(Complex c) {
    printf("%.2f + %.2fi\n", c[0], c[1]);
```

```
}
```

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 b;
    float l;
} Rectangle;
int main() {
    Rectangle r;
    float area, perimeter;
    printf("Enter breadth and length of rectangle: ");
    scanf("%f %f", &r.b, &r.l);
    printf("Area: %.2f\n", r.b*r.l);
    printf("Perimeter: %.2f\n", 2*(r.l+r.b));

    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>

int add(int a, int b);

int sub(int a, int b);

int mul(int a, int b);

int div(int a, int b);

int main() {

    int a, b, result;

    char op;

    int (*fun_ptr)(int, int) = NULL;

    printf("Enter two integers: ");

    scanf("%d %d", &a, &b);

    printf("Choose operation (+, -, *, /): ");

    scanf(" %c", &op);

    switch (op) {

        case '+':

            fun_ptr = add;

            break;
```

```
        case '-':
            fun_ptr = sub;
            break;
        case '*':
            fun_ptr = mul;
            break;
        case '/':
            fun_ptr = div;
            break;
        default:
            printf("\nError\n");
            return 1;
    }
    result = fun_ptr(a, b);
    printf("Result: %d\n", result);
    return 0;
}
int add(int a, int b) {
    return a + b;
}


int sub(int a, int b) {
    return a - b;
}
```

```c
int mul(int a, int b) {

    return a * b;

}



int div(int a, int b) {

    if (b != 0) {

        return a / b;

    } else {

        printf("Error: Division by zero\n");

        return 0;

    }

}
```

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

```c
#include <stdio.h>

int max(int arr[], int size);

int min(int arr[], int size);
```

```c
int sum(int arr[], int size);

int main() {

    int n, op, result;

    printf("Enter size of array: ");

    scanf("%d", &n);

    int arr[n];

    printf("\nEnter elements: ");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

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

    scanf("%d", &op);

    int (*fun_ptr)(int[], int) = NULL;

    switch (op) {

        case 1:

            fun_ptr = max;

            break;

        case 2:

            fun_ptr = min;

            break;

        case 3:

            fun_ptr = sum;

            break;

        default:

            printf("Error\n");
```

```
            return 1;
    }
    result = fun_ptr(arr, n);
    printf("Result: %d\n", result);

    return 0;
}
int max(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 min(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 sum(int arr[], int size) {

    int sum = 0;

    for (int i = 0; i < size; i++) {

        sum += arr[i];

    }

    return 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 onProcess();
void onEnd();
int main() {
    int op;
    void (*fun_ptr)() = NULL;
```

```
    printf("Choose event (1 for onStart, 2 for onProcess, 3 for onEnd):
");
    scanf("%d", &op);
    switch (op) {
        case 1:
            fun_ptr = onStart;
            break;
        case 2:
            fun_ptr = onProcess;
            break;
        case 3:
            fun_ptr = onEnd;
            break;
        default:
            printf("Error\n");
            return 1;
    }
    fun_ptr();


    return 0;
}
void onStart() {
    printf("Event: onStart\n");
    printf("Starting the process...\n");
}

void onProcess() {
    printf("Event: onProcess\n");
    printf("Processing the event...\n");
}

void onEnd() {
    printf("Event: onEnd\n");
    printf("Ending 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 r, int c, int m1[][c], int m2[][c], int res[][c]);
void sub(int r, int c, int m1[][c], int m2[][c], int res[][c]);
void mul(int r, int c, int m1[][c], int m2[][c], int res[][c]);
void print(int r, int c, int m[][c]);
int main() {
    int r, c, op;
    printf("Enter matrix size (rows and columns): ");
    scanf("%d %d", &r, &c);
    int m1[r][c], m2[r][c], res[r][c];

    printf("Enter first matrix:\n");
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            scanf("%d", &m1[i][j]);

    printf("Enter second matrix:\n");
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            scanf("%d", &m2[i][j]);

    printf("Choose operation (0 for Add, 1 for Subtract, 2 for
Multiply): ");
    scanf("%d", &op);
```

```
    void (*fun_ptr_arr[])(int, int, int[][c], int[][c], int[][c]) =
{add,sub,mul};


    fun_ptr_arr[op](r, c, m1,m2, res);
    printf("\nResultant matrix is:\n");
    print(r, c, res);
    return 0;
}
void add(int r, int c, int m1[][c], int m2[][c], int res[][c]) {
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            res[i][j] = m1[i][j] + m2[i][j];
        }
    }
}

void sub(int r, int c, int m1[][c], int m2[][c], int res[][c]) {
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            res[i][j] = m1[i][j] - m2[i][j];
        }
    }
}

void mul(int r, int c, int m1[][c], int m2[][c], int res[][c]) {
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            res[i][j] = 0;
            for (int k = 0; k < c; k++) {
                res[i][j] += m1[i][k] * m2[k][j];
            }
        }
    }
}

void print(int r, int c, int m[][c]) {
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            printf("%d ", m[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

```c
#include <stdio.h>
#include <string.h>
#include<stdlib.h>
typedef struct {
    int airbags : 1;
    int abs : 1;
    int sunroof : 1;
}Features;
typedef struct Vehicle {
    char manufacturer[50];
    int model_year;
    int type;
    Features features;
    union {
        struct {
```

```
            int num_doors;
            int seating_capacity;
        }car;
        struct {
            int engine_capacity;
            char bike_type[10];
        }bike;
        struct {
            int load_capacity;
            int num_axles;
        }truck;
    }details;
}Vehicle;
void add(Vehicle *v);
void display(Vehicle *v);
int main() {
    Vehicle vehicle;
    int op;

    while (1) {
        printf("\n1. Add Vehicle\n2. Display Vehicle Details\n3.
Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &op);

        switch (op) {
            case 1:
                add(&vehicle);
                break;
            case 2:
                display(&vehicle);
                break;
            case 3:
                printf("\nExiting...");
                return 0;
            default:
                printf("Error\n");
                break;
        }
    }

    return 1;
}
```

```c
void add(Vehicle *v) {
    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &v->type);
    printf("Enter manufacturer name: ");
    scanf(" %s", v->manufacturer);
    printf("Enter model year: ");
    scanf("%d", &v->model_year);

    if (v->type == 1) {
        printf("Enter number of doors: ");
        scanf("%d", &v->details.car.num_doors);
        printf("Enter seating capacity: ");
        scanf("%d", &v->details.car.seating_capacity);
    } else if (v->type == 2) {
        printf("Enter engine capacity (cc): ");
        scanf("%d", &v->details.bike.engine_capacity);
        printf("Enter bike type (e.g., sports, cruiser): ");
        scanf("%s", v->details.bike.bike_type);
    } else if (v->type == 3) {
        printf("Enter load capacity (tons): ");
        scanf("%d", &v->details.truck.load_capacity);
        printf("Enter number of axles: ");
        scanf("%d", &v->details.truck.num_axles);
    }

    int airbags, abs, sunroof;
    printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
    scanf("%d %d %d", &airbags, &abs, &sunroof);
    v->features.airbags = airbags;
    v->features.abs = abs;
    v->features.sunroof = sunroof;
}

void display(Vehicle *v) {
    printf("Manufacturer: %s\n", v->manufacturer);
    printf("Model Year: %d\n", v->model_year);

    if (v->type == 1) {
        printf("Type: Car\n");
        printf("Number of Doors: %d\n", v->details.car.num_doors);
        printf("Seating Capacity: %d\n",
v->details.car.seating_capacity);
    } else if (v->type == 2) {
```

```
        printf("Type: Bike\n");
        printf("Engine Capacity: %d cc\n",
v->details.bike.engine_capacity);
        printf("Bike Type: %s\n", v->details.bike.bike_type);
    } else if (v->type == 3) {
        printf("Type: Truck\n");
        printf("Load Capacity: %d tons\n",
v->details.truck.load_capacity);
        printf("Number of Axles: %d\n", v->details.truck.num_axles);
    }

    printf("Features: Airbags: %s, ABS: %s, Sunroof: %s\n",
           v->features.airbags ? "Yes" : "No",
           v->features.abs ? "Yes" : "No",
           v->features.sunroof ? "Yes" : "No");
}
```

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

```
#include <stdio.h>

int fact(int);

int main(){
    int n;
    printf("Enter the no to find factorial:: ");
    scanf("%d",&n);
    printf("\n");
    int f = fact(n);
    printf("\nFactorial = %d",f);
    return 0;
}

int fact(int n){

    if(n == 0){
        return 1;
    }
    int f = n * fact(n-1);
    return f;
```

```
}
```

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

```c
#include <stdio.h>

int sumdigit(int);

int main(){
    int n;
    printf("Enter a number:: ");
    scanf("%d",&n);
    printf("\n");
    int s = sumdigit(n);
    printf("\nSum of digits = %d",s);
    return 0;
}

int sumdigit(int n){
    int s=0;
    int r=n%10;
    if(n == 0){
        return 0;
    }
    s = r + sumdigit(n/10);
    return s;
}
```

3. With Recursion Find Out the maximum number in a given array

```c
#include <stdio.h>

int max(int a[],int n);

int main(){
    int n;
    printf("Enter size of array:: ");
    scanf("%d",&n);
    int a[n];
    printf("\n");
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
```

```
    int m =max(a,n);
    printf("\nLargest is = %d",m);
    return 0;
}

int max(int a[],int n){
    if (n == 1) {
        return a[0];
    }
    int m = max(a, n - 1);
    return (a[n - 1] > m) ? a[n - 1] : m;
}
```

4. With recursion calculate the power of a given number

```
#include <stdio.h>


int power(int base, int exp);


int main() {

    int base, exp;

    printf("Enter the base: ");

    scanf("%d", &base);

    printf("Enter the exponent: ");

    scanf("%d", &exp);

    int result = power(base, exp);


    printf("%d raised to the power %d is: %d\n", base, exp, result);


    return 0;

}
```

```c
int power(int base, int exp) {



    if (exp == 0) {

        return 1;

    }

    return base * power(base, exp - 1);

}
```

5. With Recursion calculate the length of a string.

```c
#include <stdio.h>


int Length(char str[], int i);


int main() {

    char str[100];


    printf("Enter the string: ");

    scanf("%s", str);


    int l = Length(str, 0);


    printf("The length of the string is: %d\n", l);


    return 0;
```

```
}

int Length(char str[], int i) {

    if (str[i] == '\0') {

        return 0;

    }

    return 1 + Length(str, i + 1);

}
```

6. With recursion reversal of a string

```
#include <stdio.h>


void reverse(char str[], int i);


int main() {

    char str[100];

    printf("Enter the string: ");

    scanf("%s", str);

    printf("\nReversed string is::");

    reverse(str, 0);



    return 0;

}

void reverse(char str[], int i) {
```

```
    if (str[i] == '\0') {

        return;

    }

    reverse(str, i + 1);

    printf("%c", str[i]);

}
```