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>

typedef struct
{
   int real;
   int imag;
}complex;

complex addcomplex(complex c1, complex c2)
{
   complex result;
   result.real = c1.real + c2.real;
```

```
    result.imag = c1.imag + c2.imag;
    return result;
}

complex multiplycomplex(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;
}




int main()

{

    complex c1,c2,s,m;
    printf("Enter first complex number (real and imaginary): ");
    scanf("%d %d", &c1.real, &c1.imag);



    printf("Enter second complex number (real and imaginary): ");
    scanf("%d %d", &c2.real, &c2.imag);

    s=addcomplex(c1,c2);
```

```
    m=multiplycomplex(c1,c2);


    printf("Sum: ");
    printf("%d + %di\n", s.real, s.imag);


    printf("Product: ");
    printf("%d + %di\n", m.real, m.imag);



    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

```
#include <stdio.h>



typedef struct {
```

```
    float width;
    float height;
} Rectangle;


float computeArea(Rectangle r)
{
    return r.width * r.height;
}


float computePerimeter(Rectangle r)
{
    return 2 * (r.width + r.height);
}

int main()
{
    Rectangle rect;


    printf("Enter width and height of the rectangle: ");
    scanf("%f %f", &rect.width, &rect.height);


    float area = computeArea(rect);
    float perimeter = computePerimeter(rect);
```

```
    printf("Area: %.2f\n", area);
    printf("Perimeter: %.2f\n", perimeter);

    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

```
#include<stdio.h>

void addition(float a, float b);
void subtraction(float a, float b);
void multiplication(float a, float b);
void division(float a, float b);




int main()
```

```
{
   char op;
   float num1,num2;
   float (*func_ptr)(float ,float);
   printf("Enter two numbers: ");
   scanf("%f %f",&num1,&num2);

   printf("Choose op (+ - * /) :");
   scanf(" %c",&op);

   switch(op)
   {
      case '+':
         func_ptr=&addition;
         (*func_ptr)(num1,num2);
         break;

      case '-':
         func_ptr=&subtraction;
         (*func_ptr)(num1,num2);
         break;


      case '*':
         func_ptr=&multiplication;
         (*func_ptr)(num1,num2);
         break;
```

```
      case '/':
         func_ptr=&division;
         (*func_ptr)(num1,num2);
         break;

   }

}



void addition(float a, float b)
{
   printf("the sum is %f",a+b);
}

void subtraction(float a, float b)
{
   printf("the diff is %f",a-b);
}

void multiplication(float a, float b)
{
   printf("the product is %f",a*b);
}

void division(float a, float b)
{
   if(b==0)
```

```
  {
     printf("Division by 0 ");


  }
  else
  {
     printf("the quotient is %0.2f",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 maximum(int arr[],int);
void minimum(int arr[],int);
void sumofelements(int arr[],int);
```

```
int main()
{
   int op;
   int size;


   printf("Enter size of array: ");
   scanf("%d",&size);

   int arr[size];

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

   printf("Choose op (1 for Max, 2 for Min, 3 for Sum) :");
   scanf("%d",&op);


   int (*func_ptr)(int[],int );

   switch(op)
   {
```

```
        case 1:
            func_ptr=&maximum;
            (*func_ptr)(arr,size);
            break;

        case 2:
            func_ptr=&minimum;
            (*func_ptr)(arr,size);
            break;



        case 3:
            func_ptr=&sumofelements;
            (*func_ptr)(arr,size);
            break;




    }

}


void maximum(int arr[],int size)
{
    int max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
```

```
            max = arr[i];
        }
    }

    printf("The max element is %d",max);
}

void minimum(int arr[],int size)
{
    int min = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }

    printf("The min element is %d",min);
}

void sumofelements(int arr[],int size)
{
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    printf("the sum 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 onProcess();

void onEnd();


int main() {
   int op;



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

   scanf("%d", &op);



   void (*event_ptr)();


   switch (op) {
```

```
        case 1:
            event_ptr = &onStart;
            (*event_ptr)();
            break;

        case 2:
            event_ptr = &onProcess;
            (*event_ptr)();
            break;

        case 3:
            event_ptr = &onEnd;
            (*event_ptr)();
            break;

        default:
            printf("Invalid event selection!\n");
            return 1;
    }



    return 0;
}

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

```
}

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

```
#include <stdio.h>



void add_matrices(int rows, int cols, int mat1[rows][cols], int mat2[rows][cols], int result[rows][cols]);
void subtract_matrices(int rows, int cols, int mat1[rows][cols], int mat2[rows][cols], int result[rows][cols]);
void multiply_matrices(int rows, int cols, int mat1[rows][cols], int mat2[rows][cols], int result[rows][cols]);


int main()
{
   int rows, cols, op;



   printf("Enter matrix size (rows and columns): ");
   scanf("%d %d", &rows, &cols);


   int mat1[rows][cols], mat2[rows][cols], result[rows][cols];



   printf("Enter first matrix:\n");
   for (int i = 0; i < rows; i++)
   {
      for (int j = 0; j < cols; j++)
      {
         scanf("%d", &mat1[i][j]);
```

```
    }
}


printf("Enter second matrix:\n");
for (int i = 0; i < rows; i++)
{
    for (int j = 0; j < cols; j++)
    {
        scanf("%d", &mat2[i][j]);
    }
}


printf("Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): ");
scanf("%d", &op);


void (*matrix_op)(int, int, int [rows][cols], int [rows][cols], int [rows][cols]);

switch (op)
{
    case 1:
        matrix_op = &add_matrices;
        (*matrix_op)(rows, cols, mat1, mat2, result);
        break;

    case 2:
```

```
            matrix_op = &subtract_matrices;
            (*matrix_op)(rows, cols, mat1, mat2, result);
            break;

        case 3:
            matrix_op = &multiply_matrices;
            (*matrix_op)(rows, cols, mat1, mat2, result);
            break;

        default:
            printf("Invalid operation selection!\n");
            return 1;
    }


    printf("Result:\n");
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
void add_matrices(int rows, int cols, int mat1[rows][cols], int mat2[rows][cols], int result[rows][cols])
{
  for (int i = 0; i < rows; i++)
  {
    for (int j = 0; j < cols; j++)
    {
      result[i][j] = mat1[i][j] + mat2[i][j];
    }
  }
}
```

```
void subtract_matrices(int rows, int cols, int mat1[rows][cols], int mat2[rows][cols], int result[rows][cols])
{
  for (int i = 0; i < rows; i++)
  {
    for (int j = 0; j < cols; j++)
    {
      result[i][j] = mat1[i][j] - mat2[i][j];
    }
  }
}
```

```
void multiply_matrices(int rows, int cols, int mat1[rows][cols], int mat2[rows][cols], int result[rows][cols])
```

```
{
  for (int i = 0; i < rows; i++)
  {
    for (int j = 0; j < cols; j++)
    {
      result[i][j] = 0;
      for (int k = 0; k < cols; k++)
      {
        result[i][j] += mat1[i][k] * mat2[k][j];
      }
    }
  }
}
```

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

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

typedef struct {
   int airbags : 1;
   int abs1 : 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[20];
        } bike;
        struct {
            int load_capacity;
            int num_axles;
        } truck;
    } details;
} Vehicle;

void addVehicle(Vehicle *v);
void displayVehicle(const Vehicle *v);

int main() {
    Vehicle vehicle;
    int option;

    while (1) {
```

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

        switch (option) {
            case 1:
                addVehicle(&vehicle);
                break;
            case 2:
                displayVehicle(&vehicle);
                break;
            case 3:
                printf("\nExiting...\n");
                return 0;
            default:
                printf("Invalid option. Please try again.\n");
        }
    }

    return 0;
}

void addVehicle(Vehicle *v) {
    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &v->type);

    printf("Enter manufacturer name: ");
    scanf(" %49s", 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(" %19s", 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);
} else {
   printf("Invalid vehicle type entered.\n");
   return;
}

printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
```

```
    scanf("%d %d %d", &v->features.airbags, &v->features.abs1, &v->features.sunroof);
}

void displayVehicle(const Vehicle *v) {
    printf("\nVehicle Details:\n");
    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);
    } else {
        printf("Unknown vehicle type.\n");
        return;
    }

    printf("Features:\n");
    printf("  Airbags: %s\n", v->features.airbags ? "Yes" : "No");
    printf("  ABS: %s\n", v->features.abs1 ? "Yes" : "No");
```

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

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

```
#include <stdio.h>

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

    int fact =factorial(num);
    printf("the factorial is %d",fact);

    return 0;
}

int factorial(int n)
{
    if(n==1)
    {
        return n;
    }
    else
    {
```

```
    return n*factorial(n-1);
  }
}
```

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

```
#include <stdio.h>

int sumofdigits(int n);

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


  int nu = sumofdigits(num);
  printf("Sum of digits: %d\n", nu);

  return 0;
}

int sumofdigits(int n)
{
  if (n == 0)
  {
    return 0;
```

```
  }
  else
  {
     return (n % 10) + sumofdigits(n / 10);
  }
}
```

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

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

```
}
 else
{
    int max = maximum(arr, size - 1);
 return (arr[size - 1] > max) ? arr[size - 1] : max;
 }


}
```

4- With recurion calculate the power of a given number

```
#include <stdio.h>

int powerof(int ,int);

int main()
{
   int num,expo;
   printf("Enter the number: ");
   scanf("%d", &num);

   printf("Enter the exponent: ");
   scanf("%d", &expo);


   int nu = powerof(num,expo);
```

```
   printf("power is: %d\n", nu);

   return 0;
}

int powerof(int num,int expo)
{
   if (expo == 0)
   {
      return 1;
   }
   else
   {
      return num*  powerof(num,expo-1);
   }
}
```

5- With Recursion calculate the length of a string.

```
#include <stdio.h>

int lengthof(char[] );

int main()
{
   char str[20];
   printf("Enter the string: ");
```

```
   scanf("%s", str);



   int l = lengthof(str);


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


   return 0;
}


int lengthof(char str[])
{
   int i=0;
   if (str[i] == '\0')
   {
      return 0;
   }
   else
   {
      i=i+1;
      return 1 + lengthof(str+i);
   }
}
```

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;
   }
   else
   {
      reverse(str, i + 1);
      printf("%c", str[i]);
   }


}
```