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 ComplexNumbers
{
   float real;
   float imag;
}complex;

void addition(complex num1, complex num2, complex * rptr);
void multiplication(complex num1, complex num2, complex * rptr);

int main()
{
   complex num1, num2, result;
   complex *rptr = &result;

   printf("Enter the first complex number (real and imaginary parts): ");
   scanf("%f %f", &num1.real, &num1.imag);

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

```
    scanf("%f %f", &num2.real, &num2.imag);


    addition(num1, num2, rptr);
    printf("Sum: %.2f + %.2fi\n", rptr->real, rptr->imag);


    multiplication(num1, num2, rptr);
    printf("Product: %.2f + %.2fi\n", rptr->real, rptr->imag);
    return 0;
}

void addition(complex num1, complex num2, complex *rptr)
{
    rptr->real = num1.real + num2.real;
    rptr->imag = num1.imag + num2.imag;
}

void multiplication(complex num1, complex num2, complex *rptr)
{
    rptr->real = (num1.real * num2.real) - (num1.imag * num2.imag);
    rptr->imag = (num1.real * num2.imag) + (num1.imag * num2.real);


}
```

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.int

Input Example:

Enter width and height of the rectangle: 5 10

Output Example:

Area: 50.00

Perimeter: 30.00

```
 #include <stdio.h>


typedef struct Rectangle
{
  float length;
  float width;
}rect;


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


   printf("Area: %.2f\n", r.width * r.length);
   printf("Perimeter: %.2f\n", 2 * (r.width + r.length));


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

int main()
{
    int a, b;
    char op;
    char choice = 'y';
    void (*fun_ptr_arr[])(int, int) = {add, sub, mul, divi};

    do {
        printf("Enter two numbers: ");
        scanf("%d %d", &a, &b);

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

        switch(op) {
            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 '/':
                if (b != 0)
                    (*fun_ptr_arr[3])(a, b);
                else
                    printf("Division by zero is not allowed!\n");
                break;
            default:
                printf("Invalid operator!!\n");
        }

        printf("Do you want to continue (y/n)? ");
        scanf(" %c", &choice);

    } while (choice == 'y' || choice == 'Y');

    printf("Goodbye!\n");
    return 0;
}

void add(int a, int b)
{
    printf("Result: %d\n", a + b);
}
```

```
void sub(int a, int b)

{

    printf("Result: %d\n", a - b);

}


void mul(int a, int b)

{

    printf("Result: %d\n", a * b);

}


void divi(int a, int b)

{

    printf("Result: %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 maximum(int *, int);

void minimum(int *, int);

void sum_of_elements(int *, int);
```

```
int main()
{
    int s, op;
    char choice;
    void (*fun_ptr_arr[])(int *, int) = {maximum, minimum, sum_of_elements};

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

    int arr[s];

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

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

        switch (op) {
            case 1:
                (*fun_ptr_arr[0])(arr, s);
                break;
            case 2:
                (*fun_ptr_arr[1])(arr, s);
                break;
            case 3:
                (*fun_ptr_arr[2])(arr, s);
                break;
            default:
                printf("Invalid option!!\n");
        }

        printf("\nDo you want to continue (y/n)? ");
        scanf(" %c", &choice);

    } while (choice == 'y' || choice == 'Y');

    printf("Goodbye!\n");
    return 0;
}

void maximum(int arr[], int s)
{
    int large = arr[0];
    for (int i = 1; i < s; i++) {
        if (large < arr[i])
            large = arr[i];
    }
    printf("Result: %d", large);
}

void minimum(int arr[], int s)
{
    int small = arr[0];
    for (int i = 1; i < s; i++) {
        if (small > arr[i])
            small = arr[i];
    }
    printf("Result: %d", small);
}
```

```
void sum_of_elements(int arr[], int s)
{
    int sum = 0;
    for (int i = 0; i < s; i++) {
        sum += arr[i];
    }
    printf("Result: %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()
{
    void (*fun_ptr_arr[3])() = { onStart, onProcess, onEnd };
    int choice;
    printf("Choose event (1 for onStart, 2 for onProcess, 3 for onEnd): ");
    scanf("%d", &choice);

    switch(choice)
    {
        case 1:
        (*fun_ptr_arr[0])();
        break;
        case 2:
        (*fun_ptr_arr[1])();
        break;
        case 3:
        (*fun_ptr_arr[2])();
        break;
        default:
        printf("Invalid option!!\n");
    }

    return 0;
}

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

void onProcess()
{
    printf("Event: onProcess\n");
    printf("Processing data...\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>
#include <stdlib.h>

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 r, int c);
```

```
int main() {
    void (*fun_ptr_arr[3])(int **mat1, int **mat2, int, int) = {add, sub, mul};

    int r, c, op;
    char continueChoice;

    do {
        printf("Enter matrix size (rows and columns): ");
        scanf("%d%d", &r, &c);

        int **mat1 = (int **)malloc(r * sizeof(int *));
        int **mat2 = (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));
        }

        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:
                (*fun_ptr_arr[2])(mat1, mat2, r, c);
                break;
            default:
                printf("Invalid option!!\n");
        }


        for (int i = 0; i < r; i++) {
            free(mat1[i]);
            free(mat2[i]);
        }
        free(mat1);
        free(mat2);

        printf("\nDo you want to perform another operation (y/n)? ");
        scanf(" %c", &continueChoice);

    } while (continueChoice == 'y' || continueChoice == 'Y');
```

```
    printf("Goodbye!\n");
    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 r, int c)
{
    int **result = (int **)malloc(r * sizeof(int *));
    for (int i = 0; i < r; i++) {
        result[i] = (int *)malloc(c * sizeof(int));
    }

    printf("Result:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            result[i][j] = 0;
            for (int k = 0; k < c; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }

    for (int i = 0; i < r; i++) {
        free(result[i]);
    }
    free(result);
}
```

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>

typedef union
{
    struct
    {
        int doors;
        int seatingCapacity;
    } car;

    struct
    {
        int engineCapacity;
        char type[20];
    } 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;
    VehicleDetails details;
    VehicleFeatures features;
    int vehicle_type;
} Vehicle;

void add_vehicle(Vehicle* var, int no_of_vehicle);
void display(Vehicle* var, int no_of_vehicle);

int main() {
    int op = 0, no_of_vehicle = 0;
    Vehicle vehicles[100];

    void (*fun_ptr_arr[2])(Vehicle*, int) = {add_vehicle, display};
```

```c
    do {
        printf("\n1. Add Vehicle\n");
        printf("2. Display Vehicle Details\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &op);

        switch(op) {
            case 1:
                (*fun_ptr_arr[0])(&vehicles[no_of_vehicle], no_of_vehicle);
                no_of_vehicle++;
                break;
            case 2:
                (*fun_ptr_arr[1])(&vehicles[0], no_of_vehicle);
                break;
            case 3:
                printf("Exiting!!\n");
                break;
            default:
                printf("Invalid option!!\n");
        }
    } while(op != 3);

    return 0;
}

void add_vehicle(Vehicle* var, int no_of_vehicle) {
    int airbag, ABS, sunroof;
    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf(" %d", &var->vehicle_type);
```

```
switch(var->vehicle_type) {
    case 1: // Car
        printf("Enter manufacturer name: ");
        scanf(" %[^\n]", var->manufacturer);
        printf("Enter model year: ");
        scanf(" %d", &var->modelYear);
        printf("Enter number of doors: ");
        scanf("%d", &var->details.car.doors);
        printf("Enter seating capacity: ");
        scanf("%d", &var->details.car.seatingCapacity);
        printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
        scanf("%d %d %d", &airbag, &ABS, &sunroof);
        var->features.airbags = airbag;
        var->features.ABS = ABS;
        var->features.sunroof = sunroof;
        break;
    case 2: // Bike
        printf("Enter manufacturer name: ");
        scanf(" %[^\n]", var->manufacturer);
        printf("Enter model year: ");
        scanf(" %d", &var->modelYear);
        printf("Enter engine capacity: ");
        scanf("%d", &var->details.bike.engineCapacity);
        printf("Enter bike type: ");
        scanf("%s", var->details.bike.type);
        printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
        scanf("%d %d %d", &airbag, &ABS, &sunroof);
        var->features.airbags = airbag;
        var->features.ABS = ABS;
        var->features.sunroof = sunroof;
```

```
            break;
        case 3: // Truck
            printf("Enter manufacturer name: ");
            scanf(" %[^\n]", var->manufacturer);
            printf("Enter model year: ");
            scanf(" %d", &var->modelYear);
            printf("Enter load capacity: ");
            scanf("%d", &var->details.truck.loadCapacity);
            printf("Enter number of axles: ");
            scanf("%d", &var->details.truck.numberOfAxles);
            printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
            scanf("%d %d %d", &airbag, &ABS, &sunroof);
            var->features.airbags = airbag;
            var->features.ABS = ABS;
            var->features.sunroof = sunroof;
            break;
        default:
            printf("Invalid vehicle type.\n");
    }
}

void display(Vehicle* var, int no_of_vehicle) {
    for (int i = 0; i < no_of_vehicle; i++) {
        printf("\nManufacturer: %s\n", var[i].manufacturer);
        printf("Model Year: %d\n", var[i].modelYear);

        switch(var[i].vehicle_type) {
            case 1: // Car
                printf("Type: Car\n");
                printf("Number of Doors: %d\n", var[i].details.car.doors);
                printf("Seating Capacity: %d\n", var[i].details.car.seatingCapacity);
        break;
    case 2: // Bike
        printf("Type: Bike\n");
        printf("Engine Capacity: %d cc\n", var[i].details.bike.engineCapacity);
        printf("Bike Type: %s\n", var[i].details.bike.type);
        break;
    case 3: // Truck
        printf("Type: Truck\n");
        printf("Load Capacity: %d tons\n", var[i].details.truck.loadCapacity);
        printf("Number of Axles: %d\n", var[i].details.truck.numberOfAxles);
        break;
    default:
        printf("Unknown vehicle type.\n");
        break;
}

printf("Features: ");
if (var[i].features.airbags)
    printf("Airbags: Yes, ");
else
    printf("Airbags: No, ");

if (var[i].features.ABS)
    printf("ABS: Yes, ");
else
    printf("ABS: No, ");

if (var[i].features.sunroof)
    printf("Sunroof: Yes\n");
else
    printf("Sunroof: No\n");
```

```
  }
}
```

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

```
#include <stdio.h>

int factorial(int);

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

  printf("The factorial of %d is %d\n", n, factorial(n));
}

int factorial(int n)
{
  if (n == 0 )
    return 1;
  else
    return n * factorial(n - 1);
}
```

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

```
#include <stdio.h>

int sum_of_digits(int);

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

  printf("The sum of digits of %d is %d\n", n, sum_of_digits(n));

  return 0;
}

int sum_of_digits(int n)
{
  if (n == 0)
    return 0;
  else
    return (n % 10) + sum_of_digits(n / 10);
}
```

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

```
#include <stdio.h>

int maximum(int *, int);

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

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

   printf("The maximum element in the array is %d",maximum(arr, s));

   return 0;
}

int maximum(int arr[], int s)
{

   if (s == 1)
      return arr[0];
   else
   {
      int max_in_rest = maximum(arr, s - 1);
      return (arr[s - 1] > max_in_rest) ? arr[s - 1] : max_in_rest;
   }

}
```

4. With recurion calculate the power of a given number

```
#include <stdio.h>

int power(int , int );

int main()
{
   int base, a, result;
   printf("Enter base number: ");
   scanf("%d", &base);
   printf("Enter power number(positive integer): ");
   scanf("%d", &a);
   result = power(base, a);
```

```
    printf("%d^%d = %d", base, a, result);
    return 0;
}

int power(int base, int a)
{
    if (a != 0)
        return (base * power(base, a - 1));
    else
        return 1;
}
```

5. With Recursion calculate the length of a string.

```
 #include <stdio.h>

int length(char *str,int i);

int main()
{
    char str[50];
    int find;

    printf("Enter The String To Find Length: ");
    scanf("%[^\n]", str);
    find=l #include <stdio.h>
#include <string.h>

void reverseString(char *str, int start, int end);

int main()
{
    char str[100];

    printf("Enter a string: ");
    scanf("%s", str);

    int length = strlen(str);
    reverseString(str, 0, length - 1);

    printf("Reversed string: %s\n", str);

    return 0;
}

void reverseString(char *str, int start, int end)
{
    if (start >= end)
    {
        return;
    }

    char temp = str[start];
    str[start] = str[end];
    str[end] = temp;

    reverseString(str, start + 1, end - 1);
}ength(str,0);
    printf("Length Of The Given String '%s' is = %d",str,find);

    return 0;
}

int length(char *str,int i)
{
    if(str[i]=='\0')
     return i;
    length(str,++i);

}
```

5. With recursion reversal of a string.

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

void reverseString(char *str, int start, int end);

int main()
{
    char str[100];

    printf("Enter a string: ");
    scanf("%s", str);

    int length = strlen(str);
    reverseString(str, 0, length - 1);

    printf("Reversed string: %s\n", str);

    return 0;
}

void reverseString(char *str, int start, int end)
{
    if (start >= end)
    {
        return;
    }
    char temp = str[start];
    str[start] = str[end];
    str[end] = temp;

    reverseString(str, start + 1, end - 1);
}
```