1. 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".

```
#include <stdio.h>

typedef struct {
  float real;
  float 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;
}

void displayComplex(Complex c) {
  if (c.imag >= 0)
    printf("%.2f + %.2fi\n", c.real, c.imag);
  else
    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 = addComplex(c1, c2);
  product = multiplyComplex(c1, c2);
```

```
    printf("Sum: ");
    displayComplex(sum);

    printf("Product: ");
    displayComplex(product);

    return 0;
}
```

2. 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.

```
#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 the 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;
}
```

3. 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.

```
#include <stdio.h>

float add(float a, float b) {
  return a + b;
}

float subtract(float a, float b) {
  return a - b;
}

float multiply(float a, float b) {
  return a * b;
}

float divide(float a, float b) {
  if (b != 0)
    return a / b;
  else {
    printf("Error! Division by zero.\n");
    return 0;
  }
}

int main() {
  float num1, num2, result;
  int choice;


  float (*operation[4])(float, float) = {add, subtract, multiply, divide};

  printf("Enter two numbers: ");
  scanf("%f %f", &num1, &num2);

  printf("Choose operation:\n");
  printf("1. Addition\n");
  printf("2. Subtraction\n");
  printf("3. Multiplication\n");
  printf("4. Division\n");
  printf("Enter your choice: ");
  scanf("%d", &choice);

  if (choice >= 1 && choice <= 4) {
```

```
        result = operation[choice - 1](num1, num2);
        printf("Result: %.2f\n", result);
    } else {
        printf("Invalid choice!\n");
    }

    return 0;
}
```

4. 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.

```
#include <stdio.h>

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;
}

int main() {
```

```
    int arr[] = {10, 20, 30, 40, 50};
    int size = sizeof(arr) / sizeof(arr[0]);
    int choice;

    int (*operation[])(int[], int) = {findMax, findMin, findSum};

    printf("Choose an operation:\n");
    printf("1. Find Maximum\n");
    printf("2. Find Minimum\n");
    printf("3. Find Sum\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    if (choice >= 1 && choice <= 3) {
        int result = operation[choice - 1](arr, size);
        if (choice == 1) {
            printf("Maximum: %d\n", result);
        } else if (choice == 2) {
            printf("Minimum: %d\n", result);
        } else if (choice == 3) {
            printf("Sum: %d\n", result);
        }
    } else {
        printf("Invalid choice!\n");
    }

    return 0;
}
```

5. 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.

```
#include <stdio.h>

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");
}

int main() {
  void (*eventHandler[])() = {onStart, onProcess, onEnd};
  int choice;

  printf("Choose event (1 for onStart, 2 for onProcess, 3 for onEnd): ");
  scanf("%d", &choice);

  if (choice >= 1 && choice <= 3) {
    eventHandler[choice - 1]();
  } else {
    printf("Invalid event choice!\n");
  }

  return 0;
}
```

6. 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.

```
#include <stdio.h>

void addMatrices(int a[3][3], int b[3][3], int result[3][3]) {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      result[i][j] = a[i][j] + b[i][j];
    }
  }
}

void subtractMatrices(int a[3][3], int b[3][3], int result[3][3]) {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      result[i][j] = a[i][j] - b[i][j];
    }
  }
}

void multiplyMatrices(int a[3][3], int b[3][3], int result[3][3]) {
  for (int i = 0; i < 3; i++) {
```

```
      for (int j = 0; j < 3; j++) {
        result[i][j] = 0;
        for (int k = 0; k < 3; k++) {
          result[i][j] += a[i][k] * b[k][j];
        }
      }
    }
}

void performOperation(void (*operation)(int[3][3], int[3][3], int[3][3]), int a[3][3], int
b[3][3], int result[3][3]) {
    operation(a, b, result);
}

void printMatrix(int matrix[3][3]) {
    for (int i = 0; i < 3; i++) {
      for (int j = 0; j < 3; j++) {
        printf("%d ", matrix[i][j]);
      }
      printf("\n");
    }
}

int main() {
    int a[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int b[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int result[3][3];
    int choice;

    printf("Choose operation:\n");
    printf("1. Add matrices\n");
    printf("2. Subtract matrices\n");
    printf("3. Multiply matrices\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    void (*operation)(int[3][3], int[3][3], int[3][3]);

    if (choice == 1) {
      operation = addMatrices;
    } else if (choice == 2) {
      operation = subtractMatrices;
    } else if (choice == 3) {
      operation = multiplyMatrices;
    } else {
      printf("Invalid choice!\n");
      return 0;
```

```
    }

    performOperation(operation, a, b, result);
    printf("Result:\n");
    printMatrix(result);

    return 0;
}
```

7. Problem Statement: Vehicle Management System
Write a C program to manage information about various vehicles. The program should demonstrate the following:
Structures: Use structures to store common attributes of a vehicle, such as vehicle type, manufacturer name, and model year.
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.
Typedefs: Define meaningful aliases for complex data types using typedef (e.g., for the structure and union types).
Bitfields: Use bitfields to store flags for vehicle features like airbags, ABS, and sunroof.
Function Pointers: Use a function pointer to dynamically select a function to display specific information about a vehicle based on its type.
Requirements
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.
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.
Provide a menu-driven interface to:
Add a vehicle.
Display vehicle details.
Exit the program.

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

typedef struct {
    int doors;
    int seatingCapacity;
} Car;
```

```
typedef struct {
   int engineCapacity;
   char type[20];
} Bike;

typedef struct {
   int loadCapacity;
   int numAxles;
} Truck;

typedef union {
   Car car;
   Bike bike;
   Truck truck;
} VehicleDetails;

typedef struct {
   char manufacturer[50];
   int modelYear;
   VehicleDetails details;
   struct {
      unsigned int airbags : 1;
      unsigned int abs : 1;
      unsigned int sunroof : 1;
   } features;
   void (*displayDetails)(struct Vehicle* v);
} Vehicle;

void displayCarDetails(Vehicle* v) {
   printf("Manufacturer: %s\n", v->manufacturer);
   printf("Model Year: %d\n", v->modelYear);
   printf("Type: Car\n");
   printf("Number of Doors: %d\n", v->details.car.doors);
   printf("Seating Capacity: %d\n", v->details.car.seatingCapacity);
   printf("Features: Airbags: %s, ABS: %s, Sunroof: %s\n",
      v->features.airbags ? "Yes" : "No",
      v->features.abs ? "Yes" : "No",
      v->features.sunroof ? "Yes" : "No");
}

void displayBikeDetails(Vehicle* v) {
   printf("Manufacturer: %s\n", v->manufacturer);
   printf("Model Year: %d\n", v->modelYear);
   printf("Type: Bike\n");
   printf("Engine Capacity: %d\n", v->details.bike.engineCapacity);
   printf("Type: %s\n", v->details.bike.type);
```

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

void displayTruckDetails(Vehicle* v) {
    printf("Manufacturer: %s\n", v->manufacturer);
    printf("Model Year: %d\n", v->modelYear);
    printf("Type: Truck\n");
    printf("Load Capacity: %d\n", v->details.truck.loadCapacity);
    printf("Number of Axles: %d\n", v->details.truck.numAxles);
    printf("Features: Airbags: %s, ABS: %s, Sunroof: %s\n",
        v->features.airbags ? "Yes" : "No",
        v->features.abs ? "Yes" : "No",
        v->features.sunroof ? "Yes" : "No");
}

void inputVehicle(Vehicle* v) {
    int vehicleType;
    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &vehicleType);
    getchar();

    printf("Enter manufacturer name: ");
    fgets(v->manufacturer, sizeof(v->manufacturer), stdin);
    v->manufacturer[strcspn(v->manufacturer, "\n")] = '\0';

    printf("Enter model year: ");
    scanf("%d", &v->modelYear);

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

    if (vehicleType == 1) {
        v->displayDetails = displayCarDetails;
        printf("Enter number of doors: ");
        scanf("%d", &v->details.car.doors);
        printf("Enter seating capacity: ");
        scanf("%d", &v->details.car.seatingCapacity);
    } else if (vehicleType == 2) {
        v->displayDetails = displayBikeDetails;
        printf("Enter engine capacity: ");
        scanf("%d", &v->details.bike.engineCapacity);
        printf("Enter bike type (sports, cruiser, etc.): ");
        getchar();
        fgets(v->details.bike.type, sizeof(v->details.bike.type), stdin);
```

```
    v->details.bike.type[strcspn(v->details.bike.type, "\n")] = '\0';
  } else if (vehicleType == 3) {
    v->displayDetails = displayTruckDetails;
    printf("Enter load capacity: ");
    scanf("%d", &v->details.truck.loadCapacity);
    printf("Enter number of axles: ");
    scanf("%d", &v->details.truck.numAxles);
  } else {
    printf("Invalid vehicle type!\n");
  }
}

int main() {
  Vehicle v;
  int choice;

  while (1) {
    printf("\n1. Add Vehicle\n");
    printf("2. Display Vehicle Details\n");
    printf("3. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
      case 1:
        inputVehicle(&v);
        break;
      case 2:
        if (v.displayDetails) {
          v.displayDetails(&v);
        } else {
          printf("No vehicle added yet!\n");
        }
        break;
      case 3:
        printf("Exiting program...\n");
        return 0;
      default:
        printf("Invalid choice. Please try again.\n");
    }
  }

  return 0;
}
```

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

```
#include <stdio.h>

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

int main() {
   int num;
   scanf("%d", &num);
   printf("%d\n", factorial(num));
   return 0;
}
```

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

```
#include <stdio.h>

int sumOfDigits(int n) {
   if (n == 0) return 0;
   return n % 10 + sumOfDigits(n / 10);
}

int main() {
   int num;
   scanf("%d", &num);
   printf("%d\n", sumOfDigits(num));
   return 0;
}
```

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

```
#include <stdio.h>

int findMax(int arr[], int size) {
   if (size == 1) {
      return arr[0];
   }
   int max = findMax(arr, size - 1);
   if (arr[size - 1] > max) {
      return arr[size - 1];
   }
   return max;
}

int main() {
```

```
    int arr[] = {12, 45, 2, 76, 34};
    int size = 5;
    printf("Maximum number is: %d\n", findMax(arr, size));
    return 0;
}
```

11. With recurion calculate the power of a given number

```
#include <stdio.h>

int power(int base, int exp) {
    if (exp == 0) return 1;
    return base * power(base, exp - 1);
}

int main() {
    int base, exp;
    scanf("%d %d", &base, &exp);
    printf("%d\n", power(base, exp));
    return 0;
}
```

12. With Recursion calculate the length of a string.

```
#include <stdio.h>

int stringLength(char str[]) {
    if (str[0] == '\0') return 0;
    return 1 + stringLength(str + 1);
}

int main() {
    char str[100];
    scanf("%s", str);
    printf("%d\n", stringLength(str));
    return 0;
}
```

13. With recursion revrsal of a string

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

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

int main() {
   int base, exp;
   scanf("%d %d", &base, &exp);
   printf("%d\n", power(base, exp));
   return 0;
}
```