1.

```c
/*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 complex{
    int real;
    int imag;
} Complex;

Complex addComplex(Complex, Complex);
Complex multiplyComplex(Complex, Complex);

int main() {
    Complex num1, num2;
    printf("Enter first complex number (real and imaginary): ");
    scanf("%d %d", &num1.real, &num1.imag);
    printf("Enter second complex number (real and imaginary): ");
    scanf("%d %d", &num2.real, &num2.imag);

    Complex sum = addComplex(num1, num2);
    Complex product = multiplyComplex(num1, num2);

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

    return 0;
}

Complex addComplex(Complex a, Complex b) {
    Complex result;
```

```
    result.real = a.real + b.real;
    result.imag = a.imag + b.imag;
    return result;
}

Complex multiplyComplex(Complex a, Complex b) {
    Complex result;
    result.real = a.real * b.real - a.imag * b.imag;
    result.imag = a.real * b.imag + a.imag * b.real;
    return result;
}
```

```
Enter first complex number (real and imaginary): 3 4
Enter second complex number (real and imaginary): 1 2
Sum: 4 + 6i
Product: -5 + 10i
```

2.

```
/*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 rectangle {
    float length;
    float width;
}Rectangle;

float rectangleArea(Rectangle);
float rectanglePerimeter(Rectangle);
```

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

    printf("Area: %.2f\n", rectangleArea(rectangle));
    printf("Perimeter: %.2f\n", rectanglePerimeter(rectangle));

    return 0;
}

float rectangleArea(Rectangle r) {
    return r.length * r.width;
}

float rectanglePerimeter(Rectangle r) {
    return 2 * (r.length + r.width);
}
```

```
Enter width and height of the rectangle: 5 10
Area: 50.00
Perimeter: 30.00
```

3.

```c
/*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 div(int, int);
```

```c
int main() {
    int num1, num2;
    char op;
    void (*fp_calc[])(int, int) = {add, sub, mul, div};

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    printf("Choose operation (+, -, *, /): ");
    scanf(" %c", &op);

    switch(op) {
        case '+':
            fp_calc[0](num1, num2);
            break;
        case '-':
            fp_calc[1](num1, num2);
            break;
        case '*':
            fp_calc[2](num1, num2);
            break;
        case '/':
            fp_calc[3](num1, num2);
            break;
        default:
            printf("Invalid operation!\n");
    }
    return 0;
}

void add(int num1, int num2) {
    printf("Result: %d\n", num1 + num2);
}

void sub(int num1, int num2) {
    printf("Result: %d\n", num1 - num2);
}

void mul(int num1, int num2) {
    printf("Result: %d\n", num1 * num2);
}

void div(int num1, int num2) {
```

```
    if (num2!= 0) {
        printf("Result: %.2f\n", (float)num1 / num2);
    } else {
        printf("Error: Division by zero!\n");
    }
}
```

```
Enter two numbers: 10 5
Choose operation (+, -, *, /): *
Result: 50
```

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.
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>

int max(int*,int);
int min(int*,int);
int sum(int*,int);

int main() {
    int size;
    int (*fp_arr[])(int*, int) = {max, min, sum};
    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]);
    }
```

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

    if (op >= 1 && op <= 3) {
        int result = fp_arr[op - 1](arr, size);
        printf("Result: %d\n", result);
    } else {
        printf("Invalid choice. Please choose a number between 1 and 3.\n");
    }



    return 0;
}

int max(int arr[], int size) {
    int max_val = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max_val) {
            max_val = arr[i];
        }
    }
    return max_val;
}

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

int sum(int arr[], int size) {
    int sum_val = 0;
    for (int i = 0; i < size; i++) {
        sum_val += arr[i];
    }
    return sum_val;
}
```

```
Enter size of array: 4
Enter elements: 10 20 30 40
Choose operation (1 for Max, 2 for Min, 3 for Sum): 3
Result: 100
```

5.

```c
/*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 (*eventHandlers[])() = {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) {
        eventHandlers[choice - 1]();
    } else {
        printf("Invalid choice. Please choose a number between 1 and 3.\n");
    }

    return 0;
}

void onStart() {
```

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

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

void onEnd() {
    printf("Event: onEnd\n");
    printf("Process completed.\n");
}
```

```
Choose event (1 for onStart, 2 for onProcess, 3 for onEnd): 1
Event: onStart
Starting the process...
```

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.
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_matrix(int, int, int*, int*);
void subtract_matrix(int,int, int*, int*);
```

```c
void multiply_matrix(int, int, int*, int*);

int main() {
    void (*fn_matrix[])(int, int, int*, int*) = {add_matrix, subtract_matrix,
multiply_matrix};
    int rows, cols;

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

        // Allocate memory for matrices
    int *matrix1 = (int *)malloc(rows * cols * sizeof(int));
    int *matrix2 = (int *)malloc(rows * cols * sizeof(int));
    if (!matrix1 || !matrix2) {
        printf("Memory allocation failed.\n");
        free(matrix1);
        free(matrix2);
        return 1;
    }

    // Input first matrix
    printf("Enter first matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", (matrix1 + i * cols + j));
        }
    }

    // Input second matrix
    printf("Enter second matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", (matrix2 + i * cols + j));
        }
    }

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

    if (choice >= 1 && choice <= 3) {
        fn_matrix[choice - 1](rows, cols, matrix1, matrix2);
    } else {
```

```
        printf("Invalid choice. Please try again.\n");
    }

    return 0;
};

void add_matrix(int r, int c, int *m1, int *m2) {
    printf("Result:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            printf("%d ", (*(m1 + i * c + j) + *(m2 +i * c + j)));
        }
        printf("\n");
    }
}

void subtract_matrix(int r, int c, int *m1, int *m2) {
    printf("Result:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            printf("%d ", (*(m1 + i * c + j) - *(m2 + i * c + j)));
        }
        printf("\n");
    }
}

void multiply_matrix(int r, int c, int *m1, int *m2) {
    if (c!= r) {
        printf("Multiplication not possible.\n");
        return;
    }
    printf("Result:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            int sum = 0;
            for (int k = 0; k < c; k++) {
                sum += *(m1 + i * c + k) * *(m2 + k * c + j);
            }
            printf("%d ", sum);
        }
        printf("\n");
    }
}
```

```
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
Result:
6 8
10 12
```

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.



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>

//enum for vehicle type
typedef enum { CAR = 1, BIKE, TRUCK } vehicle_type;

//structure to store common attributes of a vehicle, such as vehicle type,
manufacturer name, and model year
typedef struct {
    vehicle_type type;
    char manufacturer_name[50];
    int model_year;
} vehicle;
```

```c
//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
typedef union {
    struct {
        int number_of_doors;
        int seating_capacity;
    } car;
    struct {
        int engine_capacity;
        char bike_type[50];
    } bike;
    struct {
        int load_capacity;
        int number_of_axles;
    } truck;
} vehicle_details;

//bitfield to store vehicle features (e.g., airbags, ABS, sunroof)
typedef struct {
    unsigned int airbags:1;
    unsigned int abs:1;
    unsigned int sunroof:1;
} vehicle_features;

void add_vehicle(vehicle *, vehicle_details *, vehicle_features *);
void display_vehicle(vehicle *,vehicle_details *,vehicle_features *);

int main(){
    vehicle v;
    vehicle_details vd;
    vehicle_features vf;

    void (*vehicle_ptr[])(vehicle *, vehicle_details *, vehicle_features *)
={add_vehicle, display_vehicle};
    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);
        printf("\n");

        if (choice == 3) {
            printf("Exiting...\n");
            break;
        }

        if (choice >= 1 && choice <= 2) {
            vehicle_ptr[choice - 1](&v, &vd, &vf);
        } else {
            printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}

void add_vehicle(vehicle *v, vehicle_details *vd, vehicle_features *vf) {
    int type;
    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &type);
    if (type < 1 || type > 3) {
    printf("Invalid vehicle type. Please try again.\n");
    return;
    }
    v->type = type;
    printf("Enter manufacturer name: ");
    scanf("%s", v->manufacturer_name);
    printf("Enter model year: ");
    scanf("%d", &v->model_year);

    switch (v->type) {
        case CAR:
            printf("Enter number of doors: ");
            scanf("%d", &vd->car.number_of_doors);
            printf("Enter seating capacity: ");
            scanf("%d", &vd->car.seating_capacity);
            break;
        case BIKE:
            printf("Enter engine capacity: ");
            scanf("%d", &vd->bike.engine_capacity);
            printf("Enter bike type: ");
            scanf("%s", vd->bike.bike_type);
```

```
            break;
        case TRUCK:
            printf("Enter load capacity: ");
            scanf("%d", &vd->truck.load_capacity);
            printf("Enter number of axles: ");
            scanf("%d", &vd->truck.number_of_axles);
            break;
    }

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

void display_vehicle(vehicle *v, vehicle_details *vd, vehicle_features *vf) {
    printf("Manufacturer: %s\n", v->manufacturer_name);
    printf("Model Year: %d\n", v->model_year);

    switch (v->type) {
        case CAR:
            printf("Type: Car\n");
            printf("Number of Doors: %d\n", vd->car.number_of_doors);
            printf("Seating Capacity: %d\n", vd->car.seating_capacity);
            break;
        case BIKE:
            printf("Type: Bike\n");
            printf("Engine Capacity: %d\n", vd->bike.engine_capacity);
            printf("Bike Type: %s\n", vd->bike.bike_type);
            break;
        case TRUCK:
            printf("Type: Truck\n");
            printf("Load Capacity: %d\n", vd->truck.load_capacity);
            printf("Number of Axles: %d\n", vd->truck.number_of_axles);
            break;
        default:
            printf("No details to display.\n");
            break;
    }
    printf("Features: Airbags: %s, ABS: %s, Sunroof: %s\n", vf->airbags? "Yes" :
"No", vf->abs? "Yes" : "No", vf->sunroof? "Yes" : "No");
```

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

```
 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

 Manufacturer: Toyota
 Model Year: 2021
 Type: Car
 Number of Doors: 4
 Seating Capacity: 5
 Features: Airbags: Yes, ABS: Yes, Sunroof: No


 1. Add Vehicle
 2. Display Vehicle Details
 3. Exit
 Enter your choice: 3

 Exiting...
```

8.

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

#include <stdio.h>

int fact(int);
```

```c
int main() {
    int num;
    printf("Enter a number to find the factorial: ");
    scanf("%d", &num);
    int res = fact(num);
    printf("Factorial of %d is %d", num, res);
    return 0;
}

int fact(int num) {
    if (num == 0 || num == 1)
        return 1;
    else
        return num * fact(num-1);
}
```

```
Enter a number to find the factorial: 6
Factorial of 6 is 720
```

9.

```c
/*2. WAP to find the sum of digits of a number using recursion.*/

#include <stdio.h>
int digits_sum(int);

int main() {
    int num;
    printf("Enter a number to find the sum of its digits: ");
    scanf("%d", &num);
    printf("The sum of digits of the number %d is: %d\n", num, digits_sum(num));
    return 0;
}

int digits_sum(int num) {
    if (num == 0)
    {
        return 0;
    }
    int sum = 0;
    sum = num % 10 + digits_sum(num / 10);
```

```
    return sum;
}
```

```
Enter a number to find the sum of its digits: 1234
The sum of digits of the number 1234 is: 10
```

10.

```c
/*3. With Recursion Findout the maximum number in a given array*/

#include <stdio.h>
int max_func(int*,int);

int main() {
    int n;
    printf("Enter the size of 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 = max_func(arr, n);
    printf("Maximum number in the array: %d\n", max);
    return 0;
}

int max_func(int arr[], int n) {
    if(n == 0){
        return 0;
    }
    if(arr[n-1] > max_func(arr, n-1)){
        return arr[n-1];
    } else {
        return max_func(arr, n-1);
    }
}
```

```
Enter the size of the array: 5
Enter the elements of the array:
4 5 2 3 1
Maximum number in the array: 5
```

11.

```c
/*4. With recurion calculate the power of a given number*/

#include <stdio.h>
int power(int,int);

int main() {
    int num, exp;
    printf("Enter the number to find its power: ");
    scanf("%d", &num);
    printf("Enter the exponent: ");
    scanf("%d", &exp);

    printf("The result is: %d\n", power(num, exp));
    return 0;
}

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

```
Enter the number to find its power: 2
Enter the exponent: 3
The result is: 8
```

12.

```c
/*5. With Recursion calculate the length of a string.*/
```

```c
#include<stdio.h>

int length(char str[]);

int main() {
    char str[50];
    printf("Enter a string to find its length: ");
    scanf("%[^\n]", str);
    int len = length(str);
    printf("Length of the string: %d\n", len);
    return 0;
}

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

```
Enter a string to find its length: how are you?
Length of the string: 12
```

13.

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

char * reverse(char *, int);

int main() {
    char str[50];
    printf("Enter a string to reverse it: ");
    scanf("%[^\n]", str);
    int len = strlen(str);
    printf("Reversed string: %s\n", reverse(str, len));
    return 0;
}

char *reverse(char *str, int len) {
    if (len <= 1) {
        return str;
```

```
    }

    char temp = str[0];
    str[0] = str[len - 1];
    str[len - 1] = temp;

    reverse(str + 1, len - 2);

    return str;
}
```

```
Enter a string to reverse it: hello
Reversed string: olleh
```