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".

Input Example

Enter first complex number (real and imaginary): 34

Enter second complex number (real and imaginary): 12

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 %d\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;

}
```

o/p

```
Enter first complex number (real and imaginary): 34

Enter second complex number (real and imaginary): 12

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);

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);
}
```

o/p

Enter width and height of the rectangle: 5 10

Area: 50.00

Perimeter: 30.00

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.

Input Example:

Enter two numbers: 105

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 sub is %f", a-b);

}

void multiplication(float a, float b) {

printf("the mul is %f", a*b);

}

void division(float a, float b){
 if(b == 0) {
 printf("Division by 0 is not possible ");

}

else {

 printf("the division is %.2f",a/b);

}

}
```

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>

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);

}
```

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.

Input Example:

Choose event (1 for onStart, 2 for on Process, 3 for onEnd): 1

Output Example:

Event: onStart

Starting the process…

```
#include <stdio.h>

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("Ending the process...\n");
}

int main() {

    void (*eventHandlers[3])() = {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 select a valid event (1, 2, or 3).\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.

Input Example:

Enter matrix size (rows and columns): 22

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 addMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int
result[rows][cols]);

void subtractMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int
result[rows][cols]);

void multiplyMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[cols][cols], int
result[rows][cols]);

void performOperation(void (*operation)(int, int, int[rows][cols], int[rows][cols],
int[rows][cols]),
                      int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int
result[rows][cols]) {
    operation(rows, cols, matrix1, matrix2, result);
}

int main() {
    int rows, cols;

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

    int matrix1[rows][cols], matrix2[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", &matrix1[i][j]);
```

```
        }
    }

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

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

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

    switch (choice) {
        case 1:
            operation = addMatrices;
            break;
        case 2:
            operation = subtractMatrices;
            break;
        case 3:
            operation = multiplyMatrices;
            break;
        default:
            printf("Invalid choice!\n");
            return 1;
    }

    performOperation(operation, rows, cols, matrix1, matrix2, result);

    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 addMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int
result[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
```

```
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}

void subtractMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int
result[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }
}

void multiplyMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[cols][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] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
}
```

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]): 110

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>

typedef struct {
    char manufacturer[50];
    int modelYear;
    char type[10];
    int numDoors;
    int seatingCapacity;
    int features[3];
} Vehicle;

void addVehicle(Vehicle *v);
```

```
void displayVehicleDetails(Vehicle v);
void displayFeatures(int features[]);

int main() {
    Vehicle vehicle;
    int choice;
    int vehicleAdded = 0;

    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:
                addVehicle(&vehicle);
                vehicleAdded = 1;
                break;
            case 2:
                if (vehicleAdded) {
                    displayVehicleDetails(vehicle);
                } else {
                    printf("No vehicle details available. Please add a vehicle first.\n");
                }
                break;
            case 3:
                printf("Exiting the program.\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}

void addVehicle(Vehicle *v) {
    int typeChoice;

    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &typeChoice);

    switch (typeChoice) {
        case 1:
            strcpy(v->type, "Car");
            break;
```

```
        case 2:
            strcpy(v->type, "Bike");
            break;
        case 3:
            strcpy(v->type, "Truck");
            break;
        default:
            printf("Invalid vehicle type! Defaulting to 'Unknown'.\n");
            strcpy(v->type, "Unknown");
    }

    printf("Enter manufacturer name: ");
    scanf("%s", v->manufacturer);

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

    if (typeChoice == 1 || typeChoice == 3) {
        printf("Enter number of doors: ");
        scanf("%d", &v->numDoors);

        printf("Enter seating capacity: ");
        scanf("%d", &v->seatingCapacity);
    } else {
        v->numDoors = 0;
        v->seatingCapacity = 2;
    }

    printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
    scanf("%1d%1d%1d", &v->features[0], &v->features[1], &v->features[2]);

    printf("Vehicle added successfully!\n");
}

void displayVehicleDetails(Vehicle v) {
    printf("\nManufacturer: %s\n", v.manufacturer);
    printf("Model Year: %d\n", v.modelYear);
    printf("Type: %s\n", v.type);

    if (strcmp(v.type, "Car") == 0 || strcmp(v.type, "Truck") == 0) {
        printf("Number of Doors: %d\n", v.numDoors);
        printf("Seating Capacity: %d\n", v.seatingCapacity);
    }

    printf("Features: ");
    displayFeatures(v.features);
    printf("\n");
}
```

```
void displayFeatures(int features[]) {
    printf("Airbags: %s, ", features[0] ? "Yes" : "No");
    printf("ABS: %s, ", features[1] ? "Yes" : "No");
    printf("Sunroof: %s", features[2] ? "Yes" : "No");
}
```

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

```
#include <stdio.h>

int fact(int);

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);

}
```

o/p

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

9. 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;

else

}

int digits_sum(int num) { if (num == 0)

{

}

return 0;

int sum = 0;

sum num% 10+ digits_sum(num/10);

return sum;

}
```

o/p

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

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

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

```
int max_func(int*,int);

int main() {

printf("Enter the size of the array: ");

int n;

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);

}
```

o/p

Enter the size of the array: 5

Enter the elements of the array: 45231

Maximum number in the array: 5

11. 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;
}
```

o/p

Enter the number to find its power: 2

Enter the exponent: 3

The result is: 8

12. With Recursion calculate the length of a string.

```
#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);
}
```

o/p

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

13. String reverse

```
#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;

}
```

o/p

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