```
/*
typedef is a keyword: this is used to provide an alias
or a new name to an already existing data type

C syntac for typedef

typedef existing_name_of_the_data_type alias_name;

example
typedef double dbl;

*/
#include <stdio.h>

typedef int my_int;

int main()
{
   //alias name my_int has been used for declaring the variable
   my_int a = 28;

   printf("a = %d\n", a);

   return 0;
}



/*#include <stdio.h>



typedef struct{

   int day;

   int month;

   int year;

}dt;



int main()

{

  dt var = {26, 11 , 2024};
```

```
   printf("Size of var is %ld\n",sizeof(var));


   printf("Today date is %d-%d-%d",var.day,var.month,var.year);


    return 0;
}*/


/*#include <stdio.h>
typedef int* ip;
int main()
{
    int a = 20;
    ip ptr = &a;


    printf("001 a = %d\n",*ptr);


    *ptr = 30;


    printf("002 a = %d\n",*ptr);


    return 0;
}*/


//How to use typedef for array
```

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


typedef int arr[4];


int main(){
   arr t ={1,2,3,4};
   for(int i=0;i<4;i++)
   {
      printf("%d ",t[i]);
   }
   return 0;
}*/
```

Problem Statement 1:
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{
   float real;
   float imag;
}Complex;


Complex sum_of_num(Complex c1, Complex c2);
Complex prod_of_num(Complex c1, Complex c2);


int main()
{
   Complex c1,c2;
   printf("Enter first complex number (real and imaginary):");
   scanf("%f %f",&c1.real,&c1.imag);

   printf("\nEnter second complex number (real and imaginary):");
   scanf("%f %f",&c2.real,&c2.imag);

   Complex sum = sum_of_num(c1,c2);

   printf("sum is %.2f + %.2fi\n",sum.real,sum.imag);

   Complex product = prod_of_num(c1,c2);

   printf("Product is %.2f + %.2fi\n",product.real,product.imag);

}

Complex sum_of_num(Complex c1, Complex c2)
{
   Complex result;
   result.real = c1.real + c2.real;
   result.imag = c1.imag + c2.imag;
   return result;
}


Complex prod_of_num(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;
}
```

Typedef for Structures

Problem Statemen 2t:

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 rect);
float computePerimeter(Rectangle rect);

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;
}

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


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

/*Funtion Pointers */

```
#include <stdio.h>

void display(int);

int main(){
   //Declaration a pointer to the function display()
   void (*func_ptr)(int);
   //Initializing the pointer with the address of function display()
   func_ptr = &display;
   //Calling the function as well passing the parameter using function pointers
   (*func_ptr)(20);
   return 0;
}

void display(int a){
   printf("a = %d",a);
}
```

/*Array of Function pointers*/

```
#include <stdio.h>

void add(int, int);

void sub(int, int);

void mul(int, int);



int main(){



   void(*fun_ptr_arr[])(int,int) = {add,sub,mul};



   int a = 10,b = 20;

   (*fun_ptr_arr[0])(a,b);

   (*fun_ptr_arr[1])(a,b);
```

```
    (*fun_ptr_arr[2])(a,b);


    return 0;


}


void add(int a, int b){
    int sum = a+b;
    printf("sum = %d\n",sum);
}
void sub(int a, int b){
    int sub = a-b;
    printf("Sub is %d\n",sub);
}
void mul(int a, int b){
    int mul = a*b;
    printf("Mul is %d\n",mul);
}
```

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(){

    void(*fun_ptr_arr[])(int,int) = {add,sub,mul,divi};

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

    char op;
    printf("Choose operation(+, -, *, /):");
    getchar();
    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 '/':
        (*fun_ptr_arr[3])(a,b);
        break;
        default:
        printf("Invalid option!\n");
    }




    return 0;

}

void add(int a, int b){
    int sum = a+b;
    printf("sum = %d\n",sum);
}
void sub(int a, int b){
    int sub = a-b;
```

```
    printf("Sub is %d\n",sub);
}
void mul(int a, int b){
    int mul = a*b;
    printf("Mul is %d\n",mul);
}
void divi(int a, int b){
    int divi = a/b;
    printf("Div is %d\n",divi);
}
```

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 (*fun_ptr_arr[])(int [],int) = {max,min,sum};

    int size;
    printf("Enter the size of array:");
    scanf("%d",&size);
    int arr[size];
    printf("Enter the array elements: ");
    for(int i=0;i<size;i++)
    {
        scanf("%d",&arr[i]);
    }
    int op;
    printf("Choose operation (1 for Max, 2 for Min, 3 for Sum):");
    scanf("%d",&op);

    switch(op){
      case 1:
      int max = (*fun_ptr_arr[0])(arr,size);
      printf("MAximum is %d\n",max);
      break;
      case 2:
```

```
        int min = (*fun_ptr_arr[1])(arr,size);
        printf("Minimum is %d\n",min);
        break;
        case 3:
        int sum = (*fun_ptr_arr[2])(arr,size);
        printf("Sum is %d\n",sum);
        break;

        default:
        printf("Invalid option!\n");
    }

}

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

```
int min(int arr[],int s){
    int min=arr[0];
    for(int i=1;i<s;i++)
    {
        if(min > arr[i])
        {
            min = arr[i];
        }
    }
    return min;
}
int sum(int arr[],int s){
    int sum_1=0;
    for(int i=0;i<s;i++)
    {
        sum_1 += arr[i];
    }
    return sum_1;
}
```

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();
void onProcess();
void onEnd();

int main() {

    void (*eventHandlers[])() = {onStart, onProcess, onEnd};

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



    if (event >= 1 && event <= 3) {
        printf("Event: ");
        switch (event) {
            case 1:
                printf("onStart\n");
                eventHandlers[event - 1]();
                break;
            case 2:
                printf("onProcess\n");
                eventHandlers[event - 1]();
                break;
```

```
            case 3:
                printf("onEnd\n");
                eventHandlers[event - 1]();
                break;
        }

    } else {
        printf("Invalid event selection!\n");
    }

    return 0;
}



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

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

void onEnd() {
    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.

```
#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;

    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);

    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:
    1. Car: Number of doors and seating capacity.
    2. Bike: Engine capacity and type (e.g., sports, cruiser).
    3. 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:
    1. A char array for the manufacturer name.

2. An integer for the model year.
3. A union VehicleDetails for type-specific attributes.
4. A bitfield to store vehicle features (e.g., airbags, ABS, sunroof).
5. A function pointer to display type-specific details.

2. Write functions to:

    1. Input vehicle data, including type-specific details and features.
    2. Display all the details of a vehicle, including the type-specific attributes.
    3. Set the function pointer based on the vehicle type.

3. Provide a menu-driven interface to:

    1. Add a vehicle.
    2. Display vehicle details.
    3. Exit the program.

```
#include <stdio.h>
#include <string.h>
typedef union {
   struct {
      int num_door;
      int seating_capacity;
   } car;

   struct {
      int engine_capacity;
      char type[20];
   } bike;

   struct {
      int load_capacity;
```

```
        int num_axles;
    } truck;
} VehicleDetails;

typedef struct {
    unsigned int airbags : 1;
    unsigned int ABS : 1;
    unsigned int sunroof : 1;
} VehicleFeatures;

typedef struct {
    char manufact_name[50];
    int model_yr;
    VehicleDetails vehicle_info;
    VehicleFeatures features;
    int vehicle_type;
} Vehicle;

void add_vehicle(Vehicle* vehicles, int* vehicle_count);
void display_vehicles(Vehicle* vehicles, int* vehicle_count);

int main() {
    int option = 0, vehicle_count = 0;
    Vehicle vehicles[100];
```

```
// Array of function pointers
void (*fun_ptr[])(Vehicle*, int*) = {add_vehicle, display_vehicles};


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


    switch (option) {
        case 1:
            (*fun_ptr[0])(vehicles, &vehicle_count); // Call add_vehicle
            break;
        case 2:
            (*fun_ptr[1])(vehicles, &vehicle_count); // Call display_vehicles
            break;
        case 3:
            printf("Exiting!\n");
            break;
        default:
            printf("Invalid option!\n");
    }
} while (option != 3);
```

```
    return 0;
}

void add_vehicle(Vehicle* vehicles, int* vehicle_count) {
    int airbag, ABS, sunroof;

    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
    scanf("%d", &vehicles[*vehicle_count].vehicle_type);
    getchar(); // Clear newline from input buffer

    printf("Enter manufacturer name: ");
    fgets(vehicles[*vehicle_count].manufact_name, 50, stdin);
vehicles[*vehicle_count].manufact_name[strcspn(vehicles[*vehicle_count].manufact
_name, "\n")] = '\0'; // Remove newline

    printf("Enter model year: ");
    scanf("%d", &vehicles[*vehicle_count].model_yr);

    switch (vehicles[*vehicle_count].vehicle_type) {
        case 1: // Car
            printf("Enter number of doors: ");
            scanf("%d", &vehicles[*vehicle_count].vehicle_info.car.num_door);
            printf("Enter seating capacity: ");
            scanf("%d", &vehicles[*vehicle_count].vehicle_info.car.seating_capacity);
            break;
```

```
        case 2: // Bike
            printf("Enter engine capacity (cc): ");
            scanf("%d", &vehicles[*vehicle_count].vehicle_info.bike.engine_capacity);
            getchar(); // Clear newline
            printf("Enter bike type: ");
            fgets(vehicles[*vehicle_count].vehicle_info.bike.type, 20, stdin);

vehicles[*vehicle_count].vehicle_info.bike.type[strcspn(vehicles[*vehicle_count].veh
icle_info.bike.type, "\n")] = '\0'; // Remove newline
            break;
        case 3: // Truck
            printf("Enter load capacity (kg): ");
            scanf("%d", &vehicles[*vehicle_count].vehicle_info.truck.load_capacity);
            printf("Enter number of axles: ");
            scanf("%d", &vehicles[*vehicle_count].vehicle_info.truck.num_axles);
            break;
        default:
            printf("Invalid vehicle type.\n");
            return;
    }

    printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
    scanf("%d %d %d", &airbag, &ABS, &sunroof);

    vehicles[*vehicle_count].features.airbags = airbag;
    vehicles[*vehicle_count].features.ABS = ABS;
```

```
    vehicles[*vehicle_count].features.sunroof = sunroof;


    (*vehicle_count)++;
}


void display_vehicles(Vehicle* vehicles, int* vehicle_count) {
    for (int i = 0; i < *vehicle_count; i++) {
        printf("\nManufacturer: %s\n", vehicles[i].manufact_name);
        printf("Model Year: %d\n", vehicles[i].model_yr);


        switch (vehicles[i].vehicle_type) {
            case 1: // Car
                printf("Type: Car\n");
                printf("Number of Doors: %d\n", vehicles[i].vehicle_info.car.num_door);
                printf("Seating Capacity: %d\n",
vehicles[i].vehicle_info.car.seating_capacity);
                break;
            case 2: // Bike
                printf("Type: Bike\n");
                printf("Engine Capacity: %d cc\n",
vehicles[i].vehicle_info.bike.engine_capacity);
                printf("Bike Type: %s\n", vehicles[i].vehicle_info.bike.type);
                break;
            case 3: // Truck
                printf("Type: Truck\n");
                printf("Load Capacity: %d kg\n",
vehicles[i].vehicle_info.truck.load_capacity);
```

```
            printf("Number of Axles: %d\n", vehicles[i].vehicle_info.truck.num_axles);
            break;
        default:
            printf("Unknown vehicle type.\n");
            break;
    }

    printf("Features: ");
    printf("Airbags: %s, ", vehicles[i].features.airbags ? "Yes" : "No");
    printf("ABS: %s, ", vehicles[i].features.ABS ? "Yes" : "No");
    printf("Sunroof: %s\n", vehicles[i].features.sunroof ? "Yes" : "No");
  }
}
```

Classwork:

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

```
#include <stdio.h>

int factorial(int n);

int main(){
   int n;
   printf("Enter the number ");
   scanf("%d",&n);
   printf("\n");
   int fact = factorial(n);
   printf("factorial = %d",fact);
   return 0;
}

int factorial(int n){
   int fact = 1;
   //base condition
   if(n == 1){
      return 1;
   }
   //recursive call
```

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

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

```
#include <stdio.h>

int reverse(int,int);

int main()
{
   int n;
   printf("Enter number:");
   scanf("%d",&n);
   int rev = reverse(n,0);
   printf("Reversed number is %d",rev);

}

int reverse(int n,int rev)
{
   int rem=0;
   if(n == 0){
      return rev;
   }
   rem = n%10;
   rev =  rem + rev*10;
   return reverse(n/10,rev);
}
```

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

```
#include <stdio.h>

int find_max(int arr[], int n);

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

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

   int max = find_max(arr, n);
   printf("The maximum number in the array is %d\n", max);
```

```
    return 0;
}

int find_max(int arr[], int n) {
    if (n == 1) {
        return arr[0];
    }

    int max_in_rest = find_max(arr, n - 1);

    if (arr[n - 1] > max_in_rest) {
        return arr[n - 1];
    } else {
        return max_in_rest;
    }
}
```

4. With recurion calculate the power of a given number

```
 #include <stdio.h>

int power(int base, int exp);

int main() {
    int base, exp;
    printf("Enter the base: ");
    scanf("%d", &base);
    printf("Enter the exponent: ");
    scanf("%d", &exp);

    int result = power(base, exp);
    printf("%d raised to the power of %d is %d\n", base, exp, result);
    return 0;
}


int power(int base, int exp) {

    if (exp == 0) {
        return 1;
    }

    return base * power(base, exp - 1);
}
```

5. With Recursion calculate the length of a string.

```
 #include <stdio.h>
int my_strlen(char *str);
```

```
int main(){
   char str[100];
   printf("Enter the string:");
   scanf("%[^\n]",str);
   int len = my_strlen(str);
   printf("Length of the string: %d\n", len);
}

int my_strlen(char *str)
{

   if(*str == '\0')
   {
      return 0;
   }
   return 1 + my_strlen(str+1);
}
```

6. With recursion revrsal of a string

```
#include <stdio.h>


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


int main() {

   char str[100];

   printf("Enter the string: ");

   scanf("%[^\n]", str);




   int len = 0;

   while (str[len] != '\0') {

      len++;

   }
```

```
    reverse_string(str, 0, len - 1);

    printf("Reversed string: %s\n", str);
    return 0;
}

void reverse_string(char *str, int start, int end) {

    if (start >= end) {
        return;
    }

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

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