typedef

```
#include<stdio.h>
typedef int my_int;
int main(){
   my_int a = 28;
   printf("a = %d\n",a);
   return 0;
}
```

typedef in structures

```
#include<stdio.h>
typedef struct date{
   int day;
   int month;
   int year;
}dt;
int main(){
   dt var1 = {26,11,2024};
   printf("size of var1 = %ld\n",sizeof(var1));
   //var1 = {26,11,2024};
   printf("Today's date = %d-%d-%d\n",var1.day,var1.month,var1.year);
   return 0;
}
```

typedef in pointers

```
#include<stdio.h>
typedef int* intPtr;
int main(){
   int a = 20;
   intPtr ptr1 = &a;
   printf("a = %d\n",*ptr1);
   *ptr1 = 30;
   printf("a = %d",*ptr1);}
```

typedef in arrays

```
#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:
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 add(Complex num1, Complex num2);

Complex multiply(Complex num1, Complex num2);
```

```
void print(Complex num);

int main() {
    Complex num1, num2, sum, product;
    printf("Enter first complex number(real and imaginary): ");
    scanf("%f %f", &num1.real, &num1.imag);
    printf("Enter second complex number(real and imaginary): ");
    scanf("%f %f", &num2.real, &num2.imag);
    sum = add(num1, num2);
    product = multiply(num1, num2);
    printf("Sum: ");
    print(sum);

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

    return 0;
}
Complex add(Complex num1, Complex num2) {
    Complex result;
    result.real = num1.real + num2.real;
    result.imag = num1.imag + num2.imag;
    return result;
}
Complex multiply(Complex num1, Complex num2) {
    Complex result;
    result.real = (num1.real * num2.real) - (num1.imag * num2.imag);
    result.imag = (num1.real * num2.imag) + (num1.imag * num2.real);
    return result;
}
void print(Complex num) {
```

```
    if (num.imag < 0)
        printf("%.2f - %.2fi\n", num.real, -num.imag);
    else
        printf("%.2f + %.2fi\n", num.real, num.imag);
}
```

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{
    float width;
    float height;
}Rectangle;
void area(Rectangle r);
void perimeter(Rectangle r);
int main(){
    Rectangle r;
    printf("Enter the width and height of rectangle: \n");
    scanf("%f %f",&r.width,&r.height);
    area(r);
    perimeter(r);
```

```
    return 0;
}
void area(Rectangle r){
    float area = r.width*r.height;
    printf("Area: %.2f\n",area);
}
void perimeter(Rectangle r){
    float p = 2*(r.width+r.height);
    printf("Perimeter: %.2f",p);
}
```

Function pointers

```
#include<stdio.h>
void display(int);
int main(){
    //Declaring pointer to the function display()
    void (*func_ptr)(int);
    //Initializing pointer with address of function display()
    func_ptr = &display;
    //Calling function as well as passing paramter using function pointer
    (*func_ptr)(20);

}
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);
}
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 = %d\n",sub);
}
void mul(int a,int b){
  int mul = a*b;
  printf("mul = %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(*addptr)(int,int) = &add;
   void(*subptr)(int,int) = &sub;
   void(*mulptr)(int,int) = &mul;
   void(*diviptr)(int,int) = &divi;
   int a,b;
   char c;
   printf("Enter two numbers: ");
   scanf("%d %d",&a,&b);
   printf("Choose operation(+,-,*,/): ");
   scanf(" %c",&c);
   switch(c){
      case '+':
         (*addptr)(a,b);
         break;
      case '-':
         (*subptr)(a,b);
         break;
      case '*':
         (*mulptr)(a,b);
         break;
      case '/':
         (*diviptr)(a,b);
         break;
      default:
         printf("Invalid operation");
         break;
```

```
    }
}
void add(int a,int b){
    int ad = a+b;
    printf("Result: %d",ad);
}
void sub(int a, int b){
    int s = a-b;
    printf("Result: %d",s);
}
void mul(int a,int b){
    int m = a*b;
    printf("Result: %d",m);
}
void divi(int a,int b){
    float d = a/b;
    printf("Result: %.2f",d);
}
```

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 max(int arr[],int n);
void min(int arr[],int n);
```

```
void sum(int arr[],int n);
int main(){
   int n,op;
   void(*maxptr)(int arr[],int) = &max;
   void(*minptr)(int arr[],int) = &min;
   void(*sumptr)(int arr[],int) = &sum;
   printf("Enter the size of array: ");
   scanf("%d",&n);
   printf("Enter elements: ");
   int arr[n];
   for(int i=0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   printf("Choose operation(1 for max,2 for min,3 for sum): ");
   scanf(" %d",&op);
   switch(op){
      case 1:
         (*maxptr)(arr,n);
         break;
      case 2:
         (*minptr)(arr,n);
         break;
      case 3:
         (*sumptr)(arr,n);
         break;
      default:
         printf("Invalid option");
         break;
   }
}
void max(int arr[],int n){
```

```
    int max = arr[0];
    for(int i=1;i<n;i++){
        if(arr[i]>max){
            max = arr[i];
        }
    }
    printf("Result: %d",max);
}
void min(int arr[],int n){
    int min = arr[0];
    for(int i=1;i<n;i++){
        if(arr[i]<min){
            min = arr[i];
        }
    }
    printf("Result: %d",min);
}
void sum(int arr[],int n){
    int sum;
    for(int i=0;i<n;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(*startptr)() = &onStart;
   void(*processptr)() = &onProcess;
   void(*endptr)() = &onEnd;
   int op;
   printf("Choose event(1 for onStart,2 for onProcess,3 for onEnd): ");
   scanf(" %d",&op);
   switch(op){
      case 1:
         (*startptr)();
         break;
      case 2:
         (*processptr)();
         break;
      case 3:
         (*endptr)();
         break;
      default:
         printf("Invalid option");
         break;
   }
}
void onStart(){
   printf("Event: onStart\n");
```

```
    printf("Starting the process...");
}
void onProcess(){
    printf("Event: onProcess\n");
    printf("Processing the process...");
}
void onEnd(){
    printf("Event: onEnd\n");
    printf("Ending the process...");
}
```

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>

void add(int matrix1[][10], int matrix2[][10], int m, int n);  // Function to add matrices
void subtract(int matrix1[][10], int matrix2[][10], int m, int n);  // Function to subtract matrices
void multiply(int matrix1[][10], int matrix2[][10], int m, int n, int p);  // Function to multiply matrices

int main() {
   int m, n, p, op;

   void(*addptr)(int matrix1[][10], int matrix2[][10], int m, int n) = &add;
   void(*subptr)(int matrix1[][10], int matrix2[][10], int m, int n) = &subtract;
   void(*mulptr)(int matrix1[][10], int matrix2[][10], int m, int n, int p) = &multiply;

   // Get matrix dimensions
   printf("Enter matrix size (rows and columns for matrix A): ");
   scanf("%d %d", &m, &n);

   // Matrix B should have the same number of rows as the number of columns of matrix A
   printf("Enter the number of columns for matrix B: ");
   scanf("%d", &p);

   int matrix1[m][n];
   int matrix2[n][p];  // Matrix B has n rows and p columns (for valid multiplication)

   // Input first matrix (matrix A)
   printf("Enter first matrix (size %dx%d):\n", m, n);
   for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {
         scanf("%d", &matrix1[i][j]);
      }
   }
```

```
    // Input second matrix (matrix B)
    printf("Enter second matrix (size %dx%d):\n", n, p);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < p; j++) {
            scanf("%d", &matrix2[i][j]);
        }
    }

    // Choose operation
    printf("Choose operation (1 for add, 2 for subtract, 3 for multiply): ");
    scanf(" %d", &op);  // Added space before %d to consume any leftover newline character

    switch (op) {
        case 1:
            (*addptr)(matrix1, matrix2, m, n);
            break;
        case 2:
            (*subptr)(matrix1, matrix2, m, n);
            break;
        case 3:
            (*mulptr)(matrix1, matrix2, m, n, p);
            break;
        default:
            printf("Invalid option\n");
            break;
    }

    return 0;
}
```

```
void add(int matrix1[][10], int matrix2[][10], int m, int n) {
    int sum[m][n];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            sum[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
    printf("Result of addition:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", sum[i][j]);
        }
        printf("\n");
    }
}

void subtract(int matrix1[][10], int matrix2[][10], int m, int n) {
    int sub[m][n];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            sub[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }
    printf("Result of subtraction:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", sub[i][j]);
        }
        printf("\n");
    }
}
```

```
void multiply(int matrix1[][10], int matrix2[][10], int m, int n, int p) {
    int mul[m][p];  // Resultant matrix (size m x p)

    // Initialize the result matrix to 0
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < p; j++) {
            mul[i][j] = 0;
        }
    }

    // Matrix multiplication
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < p; j++) {
            for (int k = 0; k < n; k++) {
                mul[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }

    // Display result of multiplication
    printf("Result of multiplication:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < p; j++) {
            printf("%d ", mul[i][j]);
        }
        printf("\n");
    }
}
```

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

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

typedef union {
    int car_doors;
```

```
    int car_seats;
    int bike_engine;
    char bike_type[50];
    int truck_load;
    int truck_axles;
} VehicleDetails;

typedef struct {
    int type;
    char man_name[50];
    int model_year;
    int features : 3;
    VehicleDetails vd;
} Vehicle;

void add(Vehicle *v);
void display(Vehicle v);

int main() {
    Vehicle v;
    int ch;
    void(*addptr)(Vehicle*) = &add;
    void(*displayptr)(Vehicle) = &display;

    while(1) {
        printf("1. Add vehicle\n2. Display vehicle details\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch(ch) {
            case 1:
```

```
            (*addptr)(&v);
            break;
        case 2:
            (*displayptr)(v);
            break;
        case 3:
            printf("Exiting...\n");
            exit(0);
            break;
        default:
            printf("Invalid option\n");
            break;
      }
   }

   return 0;
}

void add(Vehicle *v) {
   printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
   scanf("%d", &v->type);

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

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

   if (v->type == 1) {
      printf("Enter number of doors: ");
      scanf("%d", &v->vd.car_doors);
```

```
        printf("Enter seating capacity: ");
        scanf("%d", &v->vd.car_seats);
    }
    else if (v->type == 2) {
        printf("Enter engine capacity: ");
        scanf("%d", &v->vd.bike_engine);

        printf("Enter type of bike: ");
        scanf("%s", v->vd.bike_type);
    }
    else if (v->type == 3) {
        printf("Enter load capacity: ");
        scanf("%d", &v->vd.truck_load);

        printf("Enter number of axles: ");
        scanf("%d", &v->vd.truck_axles);
    } else {
        printf("Incorrect type of vehicle\n");
    }

    printf("Enter features (Airbag[1/0], ABS[1/0], Sunroof[1/0]) as a 3-bit number (e.g., 111 for all): ");
    int tempfeature;
    scanf("%d", &tempfeature);
    if (tempfeature >= 0 && tempfeature <= 7) {
        v->features = tempfeature;
    } else {
        printf("Invalid feature input.\n");
        v->features = 0;
    }
}
```

```
void display(Vehicle v) {
    printf("\nManufacturer: %s\n", v.man_name);
    printf("Model year: %d\n", v.model_year);

    if (v.type == 1) {
        printf("Type: Car\n");
        printf("Number of Doors: %d\n", v.vd.car_doors);
        printf("Seating capacity: %d\n", v.vd.car_seats);
    } else if (v.type == 2) {
        printf("Type: Bike\n");
        printf("Engine capacity: %d\n", v.vd.bike_engine);
        printf("Bike type: %s\n", v.vd.bike_type);
    } else if (v.type == 3) {
        printf("Type: Truck\n");
        printf("Load capacity: %d\n", v.vd.truck_load);
        printf("Number of axles: %d\n", v.vd.truck_axles);
    }
    printf("Features: ");
    switch (v.features) {
        case 7:
            printf("Airbags: Yes, ABS: Yes, Sunroof: Yes\n");
            break;
        case 6:
            printf("Airbags: Yes, ABS: Yes, Sunroof: No\n");
            break;
        case 5:
            printf("Airbags: Yes, ABS: No, Sunroof: Yes\n");
            break;
        case 4:
            printf("Airbags: Yes, ABS: No, Sunroof: No\n");
```

```
            break;
        case 3:
            printf("Airbags: No, ABS: Yes, Sunroof: Yes\n");
            break;
        case 2:
            printf("Airbags: No, ABS: Yes, Sunroof: No\n");
            break;
        case 1:
            printf("Airbags: No, ABS: No, Sunroof: Yes\n");
            break;
        case 0:
            printf("Airbags: No, ABS: No, Sunroof: No\n");
            break;
        default:
            printf("Invalid feature combination\n");
            break;
    }
}
```

RECURSION

```
/*
Recursion: A Function calling itself


Basic synatx for Recursive Function


return funtion_name(args..){
   //base(exit) condition

   //recursion call funtion_name(args);
}


WAP to caluclate the sum of first N natural numbers using recurion
*/
```

```c
#include <stdio.h>

int sumNatural(int);

int main(){

   int n;

   printf("Enter The limit till which the summation of natural number should happen: ");
   scanf("%d",&n);
   printf("\n");

   int sum = sumNatural(n);

   printf("sum = %d",sum);
   return 0;
}

int sumNatural(int n){
   int res = 0;

   //base condition
   if(n == 0){
      return 0;
   }
   //recursive call

   res = n + sumNatural(n-1);

   return res;

}
```

5 + sumNatural(4) + sumNatural(3) + sumNatural(2) + sumNatural(1) + sumNatural(0)

0 -> 1 -> 2 -> 3 -> 4

15

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

```
#include<stdio.h>
int factorial(int);
int main(){
   int n;
   printf("Enter the number: ");
   scanf("%d",&n);
   int fact = factorial(n);
   printf("factorial = %d",fact);
   return 0;
}
int factorial(int n){
   //base condition
   if(n==0 || n==1){
      return 1;
   }
   //recursive call
   else
      return n*factorial(n-1);
}
```

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

```
#include<stdio.h>
int sumOfDigits(int n);
int main(){
   int n;
   printf("Enter the number: ");
   scanf("%d",&n);
   int sum = sumOfDigits(n);
   printf("sum of digits: %d",sum);
```

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

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

```
#include<stdio.h>
int findmax(int arr[],int n);
int main(){
    int n;
    printf("Enter the size of array: ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter array elements: ");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    int max = findmax(arr,n);
    printf("Maximum element in array: %d",max);
    return 0;
}
int findmax(int arr[],int n){
    if(n==1){
        return arr[0];
    }
    int max = findmax(arr+1,n-1);
```

```
    if(arr[0]>max){
        return arr[0];
    }
    else{
        return max;
    }}
```

4.With recurion calculate the power of a given number

```
#include<stdio.h>
int power(int base,int exponent);
int main(){
    int base,exponent;
    printf("Enter the base and exponent: ");
    scanf("%d %d",&base,&exponent);
    int p = power(base,exponent);
    printf("Power: %d",p);
    return 0;
}
int power(int base,int exponent){
    if(exponent==0){
        return 1;
    }
    else{
        return base*power(base,exponent-1);
    }
}
```

5.With Recursion calculate the length of a string.

```
#include<stdio.h>
int stringlength(char *str);
int main(){
      char str[100];
```

```
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    int length = stringlength(str);
    printf("Length of the string is: %d\n", length);

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

6.With recursion revrsal of a string

```
#include <stdio.h>
#include <string.h>
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);
}

int main() {
    char str[100];
```

```
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';
    int length = strlen(str);
    reverseString(str, 0, length - 1);
    printf("Reversed string: %s\n", str);

    return 0;
}
```