Typedef

```
#include <stdio.h>
typedef int my_int;
int main()
{
   my_int a = 28;
   printf("a = %d\n", a);
   return 0;
}
------------------------
//implement typedef along with structure
#include <stdio.h>
typedef struct date{
   int day;
   int month;
   int year;
}dt;
int main()
{
   dt today={26,11,24};
   printf("size od dt=%ld\n",sizeof(today));
   printf("todays date is:%d-%d-%d",today.day,today.month,today.year);
   return 0;

}
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
//implement typedef along with pointers
#include <stdio.h>
typedef int*intptr;
int main()
{
   int a=20;
   intptr ptr1;
   ptr1=&a;
   printf("a=%d\n",*ptr1);
   *ptr1=30;
   printf("a=%d",*ptr1);
   return 0;

}
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
//implement typedef along with pointers
#include <stdio.h>
typedef int arr[4];
int main()
{
   arr t={1,2,3,4};
   for(int i=0;i<4;i++){
      printf("a[i]=%d\n",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

```
//implement typedef along with pointers
//implement typedef along with pointers
#include <stdio.h>
typedef struct complex{
  int real;
  int imag;
}complexNumber;
int main()
{
  complexNumber num1;
  complexNumber num2;
  printf("enter 1st complex number\n");
  printf("enter the real and Imaginary part:");
  scanf("%d %d",&num1.real,&num1.imag);
  printf("enter 2md complex number\n");
  printf("enter the real and Imaginary part:");
  scanf("%d %d",&num2.real,&num2.imag);
  printf("addition of two complex number:\n");
  printf("%d+%di\n",num1.real+num2.real,num1.imag+num2.imag);
  printf("multiplication of two complex numbers is:\n");
  printf("%d+%di\n",(num1.real*num2.real)-
(num1.imag*num2.imag),(num1.real*num2.imag)+(num2.real*num1.imag));;

  return 0;

}
```

_-

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;
```

```
int main() {
   Rectangle rect;
   printf("Enter width and height of the rectangle: ");
   scanf("%f %f", &rect.width, &rect.height);
   float area = rect.width * rect.height;
   float perimeter = 2 * (rect.width + rect.height);

   printf("Area: %.2f\n", area);
   printf("Perimeter: %.2f\n", perimeter);

   return 0;
}
```

_-

FUNCTION POINTERS

```
#include <stdio.h>
void display(int);

int main() {

   void (*func_ptr)(int);//declaring a pointer to function dispaly()
   func_ptr=&display;//initializing thr pointer with the address of function dispaly()
   (*func_ptr)(20);//calling the function as well as passing the parameter using function pointers
   return 0;
}
void display(int a){
   printf("a=%d",a);
}
```

_-

ARRAY FUNCTION POINTERS

```
#include <stdio.h>
void add(int,int);
void sub(int,int);
void mul(int,int);
int main() {

   void(*ptr_arr[])(int,int)={add,sub,mul};
   int a=10,b=20;
   (*ptr_arr[0])(a,b);
   (*ptr_arr[1])(a,b);
   (*ptr_arr[2])(a,b);
   return 0;
}
void add(int a,int b){
   printf("%d + %d = %d\n",a,b,a+b);
}
void sub(int a,int b){
   printf("%d - %d = %d\n",a,b,a-b);
}
void mul(int a,int b){
   printf("%d * %d = %d\n",a,b,a*b);
}
```

_-

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() {
  int a,b;
  printf("enter two nubers:");
  scanf("%d %d",&a,&b);
  void(*ptr_arr[])(int,int)={add,sub,mul,divi};
  char op;
  while(1){
    printf("choose operation '+','-','*','/' :");
    scanf(" %c",&op);
    switch(op){
      case '+':
        (*ptr_arr[0])(a,b);
        break;
      case '-':
        (*ptr_arr[1])(a,b);
        break;
      case '*':
        (*ptr_arr[2])(a,b);
        break;
      case '/':
        (*ptr_arr[3])(a,b);
        break;
      default:
        printf("invalid operation\n");
        return 0;
    }
  }

}
void add(int a,int b){
  printf("%d + %d = %d\n",a,b,a+b);
}
void sub(int a,int b){
  printf("%d - %d = %d\n",a,b,a-b);
}
void mul(int a,int b){
  printf("%d * %d = %d\n",a,b,a*b);
}
void divi(int a,int b){
  printf("%d / %d = %f\n",b,a,(float)b/a);
}
```

-_-

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[]);
void min(int[]);
void sum(int[]);
int main() {
  int arr[4];
  printf("enter 4 array elements:");
  for(int i=0;i<4;i++){
    scanf("%d",&arr[i]);
  }
  void(*ptr_arr[])(int[4])={max,min,sum};
  int op;
  while(1){
    printf("choose operation '1','2','3':");
    scanf(" %d",&op);
    switch(op){
      case 1:
        (*ptr_arr[0])(arr);
        break;
      case 2:
         (*ptr_arr[1])(arr);
        break;
      case 3:
        (*ptr_arr[2])(arr);
        break;
      default:
        printf("invalid operation\n");
        return 0;
    }
  }

}
void max(int arr[]){
  int temp;
  for(int i=0;i<3;i++){
    if(arr[i]>arr[i+1]){
      temp=arr[i];
      arr[i]=arr[i+1];
      arr[i+1]=temp;
    }
  }
  printf("largest element in array is %d\n",arr[3]);
}
void min(int arr[]){
```

```
   int temp;
   for(int i=0;i<3;i++){
      if(arr[i]<arr[i+1]){
         temp=arr[i];
         arr[i]=arr[i+1];
         arr[i+1]=temp;
      }
   }
   printf("smallest element in array is %d\n",arr[3]);
}
void sum(int arr[]){
   int sum=0;
   for(int i=0;i<4;i++){
      sum=sum+arr[i];
   }
   printf("sum of element in array is %d\n",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);
void onProcess(void);
void onEnd(void);
int main() {

   void(*ptr_arr[])(void)={onStart,onProcess,onEnd};
   int op;
   while(1){
      printf("choose operation '1','2','3':");
      scanf(" %d",&op);
      switch(op){
         case 1:
            (*ptr_arr[0])();
            break;
         case 2:
             (*ptr_arr[1])();
            break;
         case 3:
            (*ptr_arr[2])();
            break;
         default:
            printf("invalid operation\n");
            return 0;
      }
   }
```

```
}
void onStart(void){
  printf("Starting the process\n");
}
void onProcess(void){
  printf("Process ongoing\n");
}
void onEnd(void){
  printf("process ended\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.

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[2][2],int[2][2]);
void sub(int[2][2],int[2][2]);
void mul(int[2][2],int[2][2]);
int main() {
  int arr1[2][2],arr2[2][2];
  printf("enter array 1 elements:\n");
  for(int i=0;i<2;i++){
    for(int j=0;j<2;j++){
      scanf("%d",&arr1[i][j]);
    }
  }
  printf("enter array 2 elements:\n");
  for(int i=0;i<2;i++){
    for(int j=0;j<2;j++){
      scanf("%d",&arr2[i][j]);
    }
  }
  void(*ptr_arr[])(int[2][2],int[2][2])={add,sub,mul};
  int op;
  while(1){
    printf("choose operation '1','2','3':");
    scanf(" %d",&op);
    switch(op){
      case 1:
        (*ptr_arr[0])(arr1,arr2);
```

```
            break;
        case 2:
            (*ptr_arr[1])(arr1,arr2);
            break;
        case 3:
            (*ptr_arr[2])(arr1,arr2);
            break;
        default:
            printf("invalid operation\n");
            return 0;
    }
  }

}
void add(int arr1[2][2],int arr2[2][2]){
  int sum=0;
  printf("sum\n");
  for(int i=0;i<2;i++){
    for(int j=0;j<2;j++){
      sum=arr1[i][j]+arr2[i][j];
      printf("%d\t",sum);
    }
    printf("\n");
  }

}
void sub(int arr1[2][2],int arr2[2][2]){
  int dif=0;
  printf("dif\n");
  for(int i=0;i<2;i++){
    for(int j=0;j<2;j++){
      dif=arr1[i][j]-arr2[i][j];
      printf("%d\t",dif);
    }
    printf("\n");
  }

}
void mul(int arr1[2][2], int arr2[2][2]) {
  int result[2][2];
  int row1 = 2, col1 = 2;
  int row2 = 2, col2 = 2;
  if (col1 != row2) {
    printf("Matrix multiplication not possible! The number of columns in the first matrix must equal the number of
rows in the second matrix.\n");
    return;
  }
  printf("Multiplication of the arrays (Matrix multiplication):\n");

  for (int i = 0; i <2; i++) {
    for (int j = 0; j <2; j++) {
      result[i][j] = 0;
      for (int k = 0; k <2; k++) {
        result[i][j] += arr1[i][k] * arr2[k][j];
      }
    }
```

```
    }

    for (int i = 0; i <2; i++) {
        for (int j = 0; j <2; j++) {
            printf("%d\t", result[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>
#include<string.h>
int no_Doors;
int seating_Capacity;
int noDoors;
int seating_Capacity;
int engine_Capacity;
char type_[20];
int load_Capacity;
int num_Of_Axles;

typedef struct features{
   int airbag:1;
   int ABS:1;
   int sunroof:1;
}F1;
typedef union specific_Attributes{
   int noDoors;
   int seatingCapacity;
   int engineCapacity;
   char type[20];
   int loadCapacity;
   int numOfAxles;
}U1;
typedef struct common_Attributes{
   char vehicle_Type[20];
   char manufaturer[20];
   int model_Year;
   U1 structXunion;
   F1 FEATURES;

}S1;
void adddata(S1*);
void displaydata(S1*);
int main(){
   int op;
   while(1){

      S1 COMMON_ATTRIBUTES;

      printf("Enter your choice\n1-Add vehicle\n2-display vehicle info\n3-exit\n");
      scanf("%d",&op);
      switch(op){
         case 1:
```

```
            adddata(&COMMON_ATTRIBUTES);
            break;
        case 2:
            displaydata(&COMMON_ATTRIBUTES);
            break;
        case 3:
            printf("exiting system\n");
            return 0;
        default:
            printf("invalid operation\n");
      }
    }
    return 0;
}
void adddata(S1 *COMMON_ATTRIBUTES){
    int v;
    printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck):");
    scanf("%d",&v);
    if(v==1){
        printf("enter vehicle details:\n");
        strcpy(COMMON_ATTRIBUTES->vehicle_Type,"car");
        printf("manufaturer:");
        scanf(" %[^\n]",COMMON_ATTRIBUTES->manufaturer);
        printf("model_Year:");
        scanf(" %d",&COMMON_ATTRIBUTES->model_Year);
        int airbag,sunroof,ABS;
        printf("no of doors:");
        scanf("%d",&no_Doors);
        printf("seating Capacity:");
        scanf("%d",&seating_Capacity);
        printf("enter extra features 1-yes,2-no\n");
        printf("airbag:sunroof:ABS\t");
        scanf("%d %d %d",&airbag,&sunroof,&ABS);
        COMMON_ATTRIBUTES->FEATURES.airbag=airbag;
        COMMON_ATTRIBUTES->FEATURES.sunroof=sunroof;
        COMMON_ATTRIBUTES->FEATURES.ABS=ABS;


    }
    else if(v==2){
        printf("enter vehicle details:\n");
        strcpy(COMMON_ATTRIBUTES->vehicle_Type,"bike");
        printf("manufaturer:");
        scanf(" %[^\n]",COMMON_ATTRIBUTES->manufaturer);
        printf("model_Year:");
        scanf(" %d",&COMMON_ATTRIBUTES->model_Year);
        printf("engine Capacity:");
        scanf("%d",&engine_Capacity);
        printf("type:");
        scanf(" %[^\n]",type_);

    }
    else if(v==3){
        printf("enter vehicle details:\n");
        strcpy(COMMON_ATTRIBUTES->vehicle_Type,"truck");
        printf("manufaturer:");
```

```
      scanf(" %[^\n]",COMMON_ATTRIBUTES->manufaturer);
      printf("model_Year:");
      scanf(" %d",&COMMON_ATTRIBUTES->model_Year);
      printf("load Capacity:");
      scanf("%d",&load_Capacity);
      printf("no of axles:");
      scanf(" %d",&num_Of_Axles);
   }

}
void displaydata(S1 *COMMON_ATTRIBUTES){
   printf("vehicle_Type:");
   printf("%s\n",COMMON_ATTRIBUTES->vehicle_Type);
   printf("manufaturer:");
   printf(" %s\n",COMMON_ATTRIBUTES->manufaturer);
   printf("model_Year:");
   printf(" %d\n",COMMON_ATTRIBUTES->model_Year);
   if(strcmp(COMMON_ATTRIBUTES->vehicle_Type,"car")==0){
      printf("no of doors:");
      COMMON_ATTRIBUTES->structXunion.noDoors=no_Doors;
      printf("%d\n",COMMON_ATTRIBUTES->structXunion.noDoors);
      printf("seating Capacity:");
      COMMON_ATTRIBUTES->structXunion.seatingCapacity=seating_Capacity;
      printf("%d\n",COMMON_ATTRIBUTES->structXunion.seatingCapacity);
      printf("airbag:%s\t",(COMMON_ATTRIBUTES->FEATURES.airbag)?"yes":"no");
      printf("sunroof:%s\t",(COMMON_ATTRIBUTES->FEATURES.sunroof)?"yes":"no");
      printf("ABS:%s\n",(COMMON_ATTRIBUTES->FEATURES.ABS)?"yes":"no");

   }
   else if(strcmp(COMMON_ATTRIBUTES->vehicle_Type,"bike")==0){
      printf("engine Capacity:");
      COMMON_ATTRIBUTES->structXunion.engineCapacity=engine_Capacity;
      printf("%d\n",COMMON_ATTRIBUTES->structXunion.engineCapacity);
      printf("type:");
      strcpy(COMMON_ATTRIBUTES->structXunion.type,type_);
      printf(" %s\n",COMMON_ATTRIBUTES->structXunion.type);
   }
    else if(strcmp(COMMON_ATTRIBUTES->vehicle_Type,"truck")==0){
      printf("load Capacity:");
      COMMON_ATTRIBUTES->structXunion.loadCapacity=load_Capacity;
      printf("%d\n",COMMON_ATTRIBUTES->structXunion.loadCapacity);
      printf("no of num Of Axles:");
      COMMON_ATTRIBUTES->structXunion.numOfAxles=num_Of_Axles;
      printf(" %d\n",COMMON_ATTRIBUTES->structXunion.numOfAxles);
   }
}
```

_-

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

```
#include <stdio.h>
int factorial(int a);
int main()
{
   int num;
   printf("enter the number");
   scanf("%d",&num);
   printf("\nfactorial of %d is %d",num,factorial(num));
```

```
   return 0;
}

int factorial(int a){
   if(a==0||a==1){
      printf("1");
      return 1;
   }
   else{
      printf("%d*",a);
      int fact=a*factorial(a-1);
      return fact;
   }
}
```

_-

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

```
#include <stdio.h>
int sum(int a);
int main()
{
   int num;
   printf("enter the number");
   scanf("%d",&num);
   printf("\nsum of digitis in %d is %d",num,sum(num));
   return 0;
}
int sum(int a){
   int s=0;
   if(a==0){
      return 0;
   }
   else{
      s=s+a%10;
      return s+sum(a/10);
   }
}
```

_-

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

```
#include <stdio.h>

// Function prototype
int findMax(int arr[], int size);

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

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

   int max = findMax(arr, n);
```

```
    printf("The maximum number in the array is: %d\n", max);

    return 0;
}


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

5. With recurion calculate the power of a given number

```
#include <stdio.h>


int power(int , int );

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

    result = power(base, e);
    printf("%d raised to the power %d is %d\n", base, e, result);
    return 0;
}
int power(int base, int e) {
    if (e == 0)
        return 1;
    else
        return base * power(base, e - 1);
}
```
-_-

6. With Recursion calculate the length of a string.

```
#include <stdio.h>


int stringLength(const char *str) {
    if (*str == '\0') /
        return 0;
    else
        return 1 + stringLength(str + 1);
}

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
```

```
   int length = stringLength(str);
   printf("The length of the string \"%s\" is %d.\n", str, length);
   return 0;
}
```

_-

7. With recursion revrsal of a string

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



void reverseString(char *, int, int);
int main() {
   char str[100];
   printf("Enter a string: ");
   scanf("%s", str);

   int length = strlen(str);
   reverseString(str, 0, length - 1);

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