DAY 15

```
/*
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;
```

1. //typedef

```
#include<stdio.h>

typedef int my_int;

int main()

{

   my_int a=28;//alias name has being used for declaringvariable

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

   return 0;


}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-1.exe'
a =28
PS D:\learning c\output>
```

2. //usecases

//1.can be used for already existing datatypes

//2.can be used foruser defined datatypes such as structures which improves the readability of code

//3.can be used with pointers

//4.can be used with array

//implementing with structure

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

```
 typedef struct date
{
   int day;
   int month;
   int year;
}dt;//dt is the alias name
int main()
{
   dt var1={26,11,24};
   printf("sizeof var1 is %d \n",sizeof(var1));
   //var1={26,11,24};//declaration and intialization at one go only
   printf("%d-%d-%d \n",var1.day,var1.month,var1.year);
   return 0;
}
```

```
PS D:\learning c\output> & .\'day15-2.exe'
sizeof var1 is 12
26-11-24
PS D:\learning c\output>
```

3.

```
/*using typedef with pointers
example

typedef int* intPtr;

intPtr ptr1;


*/
```

```
//using typedef with 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 \n",*ptr1);



    return 0;
}
```

```
PS D:\learning c\output> & .\'day15-3.exe'
a =20
a =30
PS D:\learning c\output>
```

4. //typedef for arrays

```
//typedef int arr[4]
//arr is an alias for an array of 4 int elemenys
#include<stdio.h>
typedef int arr[4];
int main()
{
    arr t={1,2,3,4};
    for(int i=0;i<4;i++)
    {
        printf("%d \n",t[i]);
    }
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-4.exe'
1
2
3
4
PS D:\learning c\output>
```

5. 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
{
   int real;
   int imag;
}complex;
int main()
{
   complex a,b ,sum,product;
   printf("enter real part and imaginery part of first number \n");
   scanf("%d %d",&a.real,&a.imag);
   printf("enter real part and imaginery part of second number \n");
   scanf("%d %d",&b.real,&b.imag);
   sum.real=a.real+b.real;
   product.real=a.real*b.real;
   sum.imag=a.imag+b.imag;
   product.imag=a.imag*b.imag;
   printf("sum is :%d+%di \n",sum.real,sum.imag);
   printf("product is :%d+%di \n",product.real,product.imag);
```

```
    return 0;
}
```

```
enter real part and imaginery part of first number
3 4
enter real part and imaginery part of second number
1 2
sum is :4+6i
product is :3+8i
PS D:\learning c\output>
```

6. **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 rect
{
    float width;
    float height;

}rectangle;
int main()
{
    rectangle r;
    printf("enter width and heught of rectangle \n");
    scanf("%f %f",&r.width,&r.height);
    printf("area:%.2f \n",r.height*r.width);
```

```
    printf("perimeter is: %.2f \n",2*(r.width+r.height));
}
```

```
enter width and heught of rectangle
5 10
area:50.00
perimeter is: 30.00
PS D:\learning c\output>
```

7. //function pointers

```
#include<stdio.h>
void display(int);
int main()
{
    //declaration a pointer to the function display
    void (*func_ptr)(int);//=&display;
    func_ptr=&display;
    (*func_ptr)(20);

    return 0;
}
void display(int a)
{
    printf("a=%d",a);
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-7.exe'
a=20
PS D:\learning c\output>
```

8. //array of function pointers

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

```
void mul(int,int);
//pointer pointing to different functions use array of pointers
int main()
{
   void (*fun_ptr_arr[])(int,int)={add,sub,mul};
   int a=10,b=20;
   (*fun_ptr_arr[0])(a,b);//add is at 0th index
   (*fun_ptr_arr[1])(a,b);//sub is at 1st index
   (*fun_ptr_arr[2])(a,b);//mul is at 2 index

}
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);
}
```

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-8.exe'
sum=30
sub=-10
mul=200
PS D:\learning c\output>
```

9. **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

```
//Simple Calculator Using Function Pointers
#include<stdio.h>
void add(int,int);
void sub(int,int);
void mul(int,int);
void div(int,int);

int main()
{
  void (*fun_ptr_arr[])(int,int)={add,sub,mul,div};
  int a,b;
  printf("enter two numbers \n");
  scanf("%d %d",&a,&b);
  char choice;
  printf("choose an option (+,-,*,/)\n");
  scanf(" %c",&choice);
  switch (choice)
```

```
  {
   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:
   break;
  }

}
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);
}
void div(int a,int b)
{
   int div=a/b;
   if(b==0)
   {
      printf("division by zero \n");
   }
   else
   {
      printf("div=%d \n",div);


   }
}
```

```
PS D:\learning c\output> & .\'day15-9.exe'
enter two numbers
5 10
choose an option (+,-,*,/)
+
sum=15
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-9.exe'
enter two numbers
5 10
choose an option (+,-,*,/)
-
sub=-5
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-9.exe'
enter two numbers
5 10
choose an option (+,-,*,/)
*
mul=50
```

```
PS D:\learning c\output> & .\'day15-9.exe'
enter two numbers
5 10
choose an option (+,-,*,/)
/
div=0
PS D:\learning c\output>
```

10.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

```
//array operations using function pointers
#include<stdio.h>
void max(int [],int);
void min(int[],int);
void sum(int [],int);
int main()
{
   int n;
   printf("enter the size of the array \n");
   scanf("%d",&n);
   int arr[n];

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

   }
   void (*fun_ptr_arr[])(int[],int)={max,min,sum};
   int choice;
   printf("choose an option( 1 for max ,2 for min , 3 for sum) \n");
   scanf("%d",&choice);
   switch (choice)
  {
   case 1:
   (*fun_ptr_arr[0])(arr,n);

   break;
   case 2:
   (*fun_ptr_arr[1])(arr,n);
```

```
    break;
    case 3:
    (*fun_ptr_arr[2])(arr,n);
    break;

   default:
    break;
   }
}
void max(int arr[],int n)
{
    int max=arr[0];
    for(int i=0;i<n;i++)
    {
        if(arr[i]>=max)
        {
            max=arr[i];
        }
    }
    printf("result:%d\n",max);
}
void min(int arr[],int n)
{
    int min=arr[0];
    for(int i=0;i<n;i++)
    {
        if(arr[i]<=min)
        {
            min=arr[i];
        }
```

```
    }
    printf("result:%d\n",min);
}
void sum(int arr[],int n)
{
    int total=0;
    for(int i=0;i<n;i++)
    {
        total+=arr[i];
    }
    printf("result:%d \n",total);
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-10.exe'
enter the size of the array
2
enter the elements:
1 2
choose an option( 1 for max ,2 for min , 3 for sum)
1
result:2
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-10.exe'
enter the size of the array
2
enter the elements:
1 2
choose an option( 1 for max ,2 for min , 3 for sum)
2
result:1
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-10.exe'
enter the size of the array
2
enter the elements:
1 2
choose an option( 1 for max ,2 for min , 3 for sum)
3
result:3
PS D:\learning c\output>
```

11.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...

```
//event system using function pointers
#include<stdio.h>
void onstart(void);
void onprocess(void);
void onend(void);
int main()
{
  void (*eventHandlers[])() = {onstart, onprocess, onend};
  int choice;
  printf("enter a choice (1 for onstart, 2 for onprocess,3 foronend )\n");
  scanf("%d",&choice);
  switch (choice)
  {
   case 1:
      (*eventHandlers[0])();
      break;
   case 2:
      (*eventHandlers[1])();
      break;
   case 3:
      (*eventHandlers[2])();
      break;
```

```
  default:
   break;
  }

}
void onstart()
{
   printf("Event:onStart \n");
   printf("starting the process \n");
}
void onprocess()
{
   printf("event:onprocess\n");
}
void onend()
{
   printf("event:onend \n");
   printf("ending the process \n");
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-11.exe'
enter a choice (1 for onstart, 2 for onprocess,3 foronend )
1
Event:onStart
starting the process
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-11.exe'
enter a choice (1 for onstart, 2 for onprocess,3 foronend )
2
event:onprocess
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-11.exe'
enter a choice (1 for onstart, 2 for onprocess,3 foronend )
3
event:onend
ending the process
PS D:\learning c\output>
```

12.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>

// Function declarations with correct parameter types
void add(int r, int c, int matrix1[r][c], int matrix2[r][c]);
void sub(int r, int c, int matrix1[r][c], int matrix2[r][c]);
void multiply(int r, int c, int matrix1[r][c], int matrix2[r][c]);

int main() {
    int r, c;
    printf("Enter the matrix size:\n");
    scanf("%d %d", &r, &c);

    int matrix1[r][c];
    int matrix2[r][c];

    // Input first matrix
    printf("Enter the first matrix:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }

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

```
    }

    // Array of function pointers
    void (*fun_ptr_arr[])(int, int, int[r][c], int[r][c]) = {add, sub, multiply};

    int choice;
    printf("Enter an option (1 for add, 2 for subtract, 3 for multiply):\n");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            fun_ptr_arr[0](r, c, matrix1, matrix2);
            break;
        case 2:
            fun_ptr_arr[1](r, c, matrix1, matrix2);
            break;
        case 3:
            fun_ptr_arr[2](r, c, matrix1, matrix2);
            break;
        default:
            printf("Invalid choice.\n");
            break;
    }

    return 0;
}

// Function to add two matrices
void add(int r, int c, int matrix1[r][c], int matrix2[r][c]) {
    int result[r][c];
    for (int i = 0; i < r; i++) {
```

```
        for (int j = 0; j < c; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
    printf("Result of Addition:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
}

// Function to subtract two matrices
void sub(int r, int c, int matrix1[r][c], int matrix2[r][c]) {
    int result[r][c];
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            result[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }
    printf("Result of Subtraction:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
}

// Function to multiply two matrices
```

```
void multiply(int r, int c, int matrix1[r][c], int matrix2[r][c]) {
    int result[r][c];
    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] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
    printf("Result of Multiplication:\n");
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
}
```

```
1 2
3 4
Enter the second matrix:
5 6
7 8
Enter an option (1 for add, 2 for subtract, 3 for multiply):
1
Result of Addition:
6 8
10 12
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-12.exe'
Enter the matrix size:
2 2
Enter the first matrix:
1 2
3 4
Enter the second matrix:
5 6
7 8
Enter an option (1 for add, 2 for subtract, 3 for multiply):
2
Result of Subtraction:
-4 -4
-4 -4
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-12.exe'
Enter the matrix size:
2 2
Enter the first matrix:
1 2
3 4
Enter the second matrix:
5 6
7 8
Enter an option (1 for add, 2 for subtract, 3 for multiply):
3
Result of Multiplication:
19 22
43 50
PS D:\learning c\output>
```

13. **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>

typedef union VechicleDetails
{
   int car_doors;
   int car_seating;
   int bike_engine;
   char bike_type[10];
   int truck_capacity;
   int truck_axles;
}vd;
typedef struct Vechicle
{
```

```
    char name[10];
    int year;
    int vechicletype;
    vd vd1;
    unsigned int airbags:1;
    unsigned int ABS:1;
    unsigned int sunroof:1;




} v;
void vechicledata(v *Vechicle);
void display_vechicle(v *Vechicle);

int main()
{



    void (*fun_ptr_arr[])(v*)={vechicledata,display_vechicle};
    v v1;
    int choice;
    while(1)
    {
        printf("1.add vechicle \n");
        printf("2.display vechicle details \n");
        printf("3.exit\n");
        printf("enter your choice \n");
        scanf("%d",&choice);
        switch (choice)
        {
        case 1:
```

```
            (*fun_ptr_arr[0])(&v1);


           break;
        case 2:
           (*fun_ptr_arr[1])(&v1);
           break;


        case 3:
          return 0;
          break;


        default:
           break;
        }
    }


}
void vechicledata(v *v1)
{


   printf("enter manufacture name \n");
   scanf("%s",v1->name);
   printf("enter manufacturer year \n");
   scanf("%d",&v1->year);
   printf("enter vechicle type (1.car ,2.bike,3.truck) \n");
   scanf("%d",&v1->vechicletype);
   int air,abs,sun;
   switch (v1->vechicletype)
   {
   case 1:
    printf("enter the no.of doors \n");
```

```
    scanf("%d",&v1->vd1.car_doors);
    printf("enter seating capacity \n");
    scanf("%d",&v1->vd1.car_seating);
    printf("enter features (airbags[1/0],ABS[1/0],sunroof[1/0])\n");
    scanf("%d %d %d",&air,&abs,&sun);
    air=v1->airbags;
    abs=v1->ABS;
    sun=v1->sunroof;
    break;
    case 2:
    printf("enter the engine capacity \n");
    scanf("%d",&v1->vd1.bike_engine);
    printf("enter bike type \n");
    scanf("%s",v1->vd1.bike_type);
    break;
    case 3:
    printf("enter the load capacity \n");
    scanf("%d",&v1->vd1.truck_capacity);
    printf("enter number of axles \n");
    scanf("%d",&v1->vd1.truck_axles);
    break;

   default:
    break;
   }



}
void display_vechicle(v *v1)
{
   printf("manufacturer: %s\n",v1->name);
```

```
printf("year: %d\n",v1->year);
switch (v1->vechicletype)
{
  case 1:
     printf("Vehicle Type: Car\n");
      printf("Number of Doors: %d\n", v1->vd1.car_doors);
      printf("Seating Capacity: %d\n", v1->vd1.car_seating);
      break;
  case 2:
    printf("Vehicle Type: Bike\n");
    printf("Engine Capacity: %d cc\n", v1->vd1.bike_engine);
    printf("Bike Type: %s\n", v1->vd1.bike_type);
    break;
 case 3:
    printf("Vehicle Type: Truck\n");
    printf("Load Capacity: %d tons\n",v1->vd1.truck_capacity);
    printf("Number of Axles: %d\n", v1->vd1.truck_axles);
    break;
 default:

    break;
}
printf("features \n");
if(v1->airbags==1)
{
 printf("airbags:yes \n");
}
else if(v1->airbags==0)
{
 printf("airbags:no \n");
}
```

```
  if(v1->ABS==1)
  {
   printf("ABS:yes \n");
  }
  else if(v1->ABS==0)
  {
   printf("ABS:no \n");
  }
  if(v1->sunroof==1)
  {
   printf("sunroof:yes \n");
  }
  else if(v1->sunroof==0)
  {
   printf("sunroof:no \n");
  }
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-13.exe'
1.add vechicle
2.display vechicle details
3.exit
enter your choice
1
enter manufacture name
maruti
enter manufacturer year
2020
enter vechicle type (1.car ,2.bike,3.truck)
1
enter the no.of doors
4
enter seating capacity
5
enter features (airbags[1/0],ABS[1/0],sunroof[1/0])
1 1 0
1.add vechicle
2.display vechicle details
3.exit
enter your choice
2
manufacturer: maruti
year: 2020
Vehicle Type: Car
Number of Doors: 5
Seating Capacity: 5
features
airbags:no
ABS:no
sunroof:no
```

15. //recursion: a function calling itself

// synatx: return functname(args..){

// base condition/exit condition-to avoid stack overflow

// recusrion call funcname(args);

// }

//wap to calculate sum of first n natural numbers using recursion

```
#include<stdio.h>
int sumNatural(int);
int main()
{
   int n;
   printf("enter the limit \n");
   scanf("%d",&n);
   printf("\n");
   int sum=sumNatural(n);
   printf("sum=%d \n",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;
}
//stack:
//0
//1
//2
//3
//4
```

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-14.exe'
enter the limit
5

sum=15
PS D:\learning c\output>
```

15. //wap to find factorial using recursion

```
int factorial(int);
int main()
{
   int n;
   printf("enter the limit \n");
   scanf("%d",&n);
   printf("\n");
   int f=factorial(n);
   printf("factorial=%d \n",f);
   return 0;

}
int factorial(int n)
{
   int fact=1;
   //base condition
   if(n==0)
   {
      return 1;
   }
   //recursive call
   fact=n*factorial(n-1);
```

```
    return fact;
}
```

```
PS D:\learning c\output> & .\'day15-15.exe'
enter the limit
4

factorial=24
PS D:\learning c\output>
```

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

```
//wap to find sum of digits of a number using recursion
#include<stdio.h>
int sum_dig(int);
int main()
{
    int n;
    printf("enter a number\n");
    scanf("%d",&n);
    int dig=sum_dig(n);
    printf("sum of digits is %d \n",dig);
}
int sum_dig(int n)
{
    // digit=n%10
    //n=n/10;
    if(n==0)
    {
        return 0;
    }
    return(n%10)+sum_dig(n/10);
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-16.exe'
enter a number
10
sum of digits is 1
PS D:\learning c\output>
```

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

```
//findout the maximum number in a given array
#include<stdio.h>
int max_array(int [],int);
int main()
{
   int n;
   printf("enter the size of array \n");
   scanf("%d",&n);
   int arr[n];
   printf("enter elements of array \n");
   for(int i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
   int maxn=max_array(arr,n);
   printf("max number %d \n",maxn);
}
int max_array(int arr[],int n)
{
   if(n==1)
   {
      return arr[0];
   }
   int max=max_array(arr,n-1);

   if(arr[n-1]>max)
```

```
    {
        return arr[n-1];
    }
    else
    {
        return max;
    }
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-17.exe'
enter the size of array
3
enter elements of array
1
2
3
max number 3
PS D:\learning c\output>
```

18. With recurion calculate the power of a given number

```
//find power of a number using recursion
#include<stdio.h>
int power(int,int);
int main()
{
    int exponent,base;
    printf("enter the base \n");
    scanf("%d",&base);
    printf("enter the power \n");
    scanf("%d",&exponent);
    int p=power(base,exponent);
    printf("power is %d \n",p);

}
```

```
int power(int base,int exponent)
{
   if(exponent==0)
   {
      return 1;
   }
   return base*power(base,exponent-1);
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day15-18.exe'
enter the base
2
enter the power
3
power is 8
```

19.With Recursion calculate the length of a string.

```
//wap to find length of a string using recursion
#include <stdio.h>

int stringLength(char str[], int index);

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

   int length = stringLength(str, 0);
   printf("Length of the string = %d\n", length);

   return 0;
}
```

```
int stringLength(char str[], int index) {
    if (str[index] == '\0') {
        return 0;
    }
    // Recursive call
    return 1 + stringLength(str, index + 1);
}
```

```
PS D:\learning c\output> & .\'day15-19.exe'
Enter a string: rinta
Length of the string = 5
PS D:\learning c\output>
```

20. With recursion revrsal of a string

```
#include <stdio.h>

void reverseString(char str[], int start, int end);

int main() {
    char str[100];
    printf("Enter a string: ");
    gets(str);

    int length = 0;
    while (str[length] != '\0') {
        length++;
    }

    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);
}
```

```
PS D:\learning c\output> & .\'day15-20.exe'
Enter a string: rinta
Reversed string: atnir
PS D:\learning c\output>
```