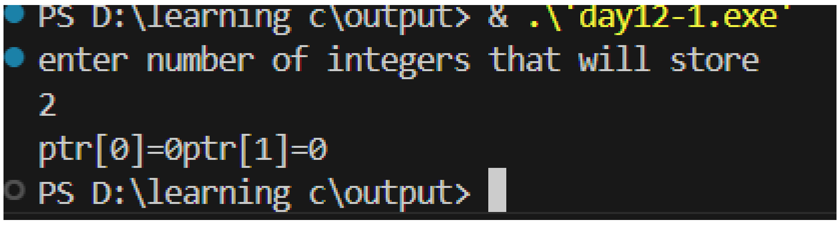
Day 12

1. #include<stdio.h>

```
#include<stdlib.h>
int main()
{
   int *ptr=NULL;
   int n;
   printf("enter number of integers that will store \n");
   scanf("%d",&n);
   ptr=(int*)calloc(n,sizeof(int));
   for(int i=0;i<n;i++)
   {
      printf("ptr[%d]=%d",i,ptr[i]);//check result by default it is zero in calloc
      //malloc always it mightnot be zero
   }
}
```

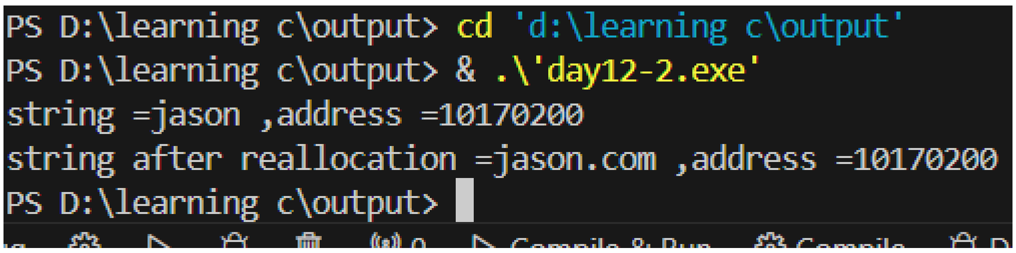


```
PS D:\learning c\output> & .\'day12-1.exe'
enter number of integers that will store
2
ptr[0]=0ptr[1]=0
PS D:\learning c\output>
```

2. //realloc

```
//reuse or extend memory that you previously allocated using malloc or calloc
//two arguments
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
   char *str;//make it constant to avoid memory leak but realloc can't be used
   str=(char*)malloc(15);//pointing to starting address of allcated memory
   strcpy(str,"jason");
```

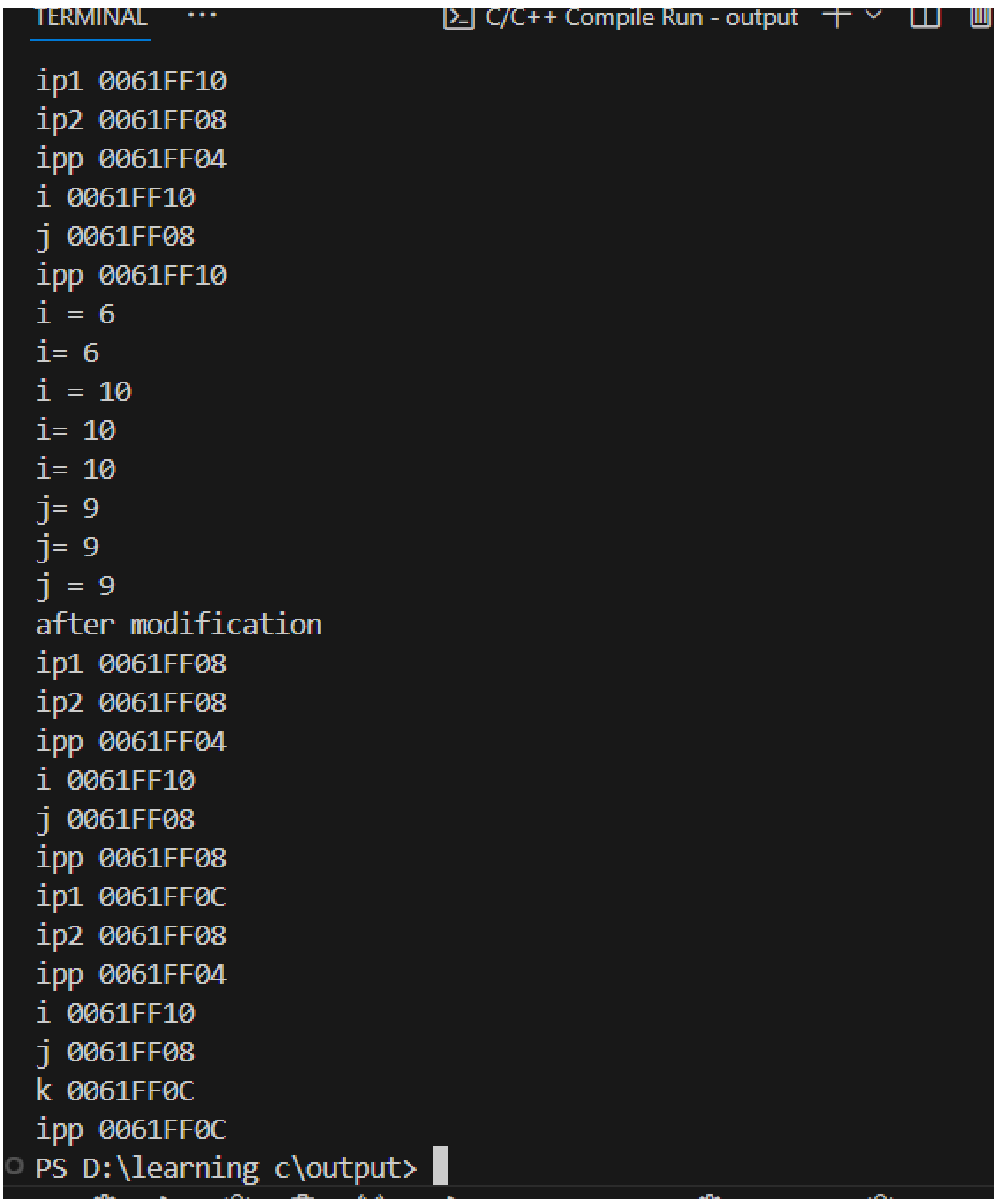
```
    printf("string =%s ,address =%u \n",str,str);
    //reallocating memory(extending)
    str=(char*)realloc(str,25);//pointing to starting address of reallocated memory
    strcat(str,".com");
    printf("string after reallocation =%s ,address =%u \n",str,str);
    free(str);
    return 0;


}
```



```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day12-2.exe'
string =jason ,address =10170200
string after reallocation =jason.com ,address =10170200
PS D:\learning c\output>
```

3. //double pointers

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int **ipp;
    int ***ipp1;
    int i=6,k=1,j=9;
    int *ip1,*ip2;
    ip1=&i;
    ip2=&j;
    ipp=&ip1;


    ipp1=&ipp;
    printf("before modification \n");
    printf("ip1 %p \n",ip1);
    printf("ip2 %p \n",ip2);
    printf("ipp %p \n",ipp);

    printf("i %p \n",&i);

    printf("j %p \n",&j);

    printf("ipp %p \n",*ipp);//adress of ip1


    printf("i = %d \n",*ip1);

    printf("i= %d \n",**ipp);

    **ipp=8;//two level cz defrencing a value from a pointer

    ***ipp1=10;

    printf("i = %d \n",*ip1);


    printf("i= %d \n",**ipp);

    printf("i= %d \n",***ipp1);

    //after modification

    *ipp=ip2;//pointing to ip2 // here one * cz level of pointing =1;//adress inside ip2 will be given that
is adress of j

    printf("j= %d \n",*ip2);

    printf("j= %d \n",**ipp);

     printf("j = %d \n",*ip1);//earlier ipp pointed to ip1 now pointed to ip2 now waht happens is

     //ip1 also pointed to j so value inside ip1 will be adress ofj

    printf("after modification \n");

    printf("ip1 %p \n",ip1);

    printf("ip2 %p \n",ip2);

    printf("ipp %p \n",ipp);

    printf("i %p \n",&i);

    printf("j %p \n",&j);

    printf("ipp %p \n",*ipp);//adress of ip1


    *ipp=&k;// here value of ipp will be adress of ip1 but value of ip1 will be address of k

    printf("ip1 %p \n",ip1);

    printf("ip2 %p \n",ip2);
    printf("ipp %p \n",ipp);

    printf("i %p \n",&i);

    printf("j %p \n",&j);

    printf("k %p \n",&k);


    printf("ipp %p \n",*ipp);



}
```



```
ip1 0061FF10
ip2 0061FF08
ipp 0061FF04
i 0061FF10
j 0061FF08
ipp 0061FF10
i = 6
i= 6
i = 10
i= 10
i= 10
j= 9
j= 9
j = 9
after modification
ip1 0061FF08
ip2 0061FF08
ipp 0061FF04
i 0061FF10
j 0061FF08
ipp 0061FF08
ip1 0061FF0C
ip2 0061FF08
ipp 0061FF04
i 0061FF10
j 0061FF08
k 0061FF0C
ipp 0061FF0C
PS D:\learning c\output>
```

Problem 1: Dynamic Array Resizing

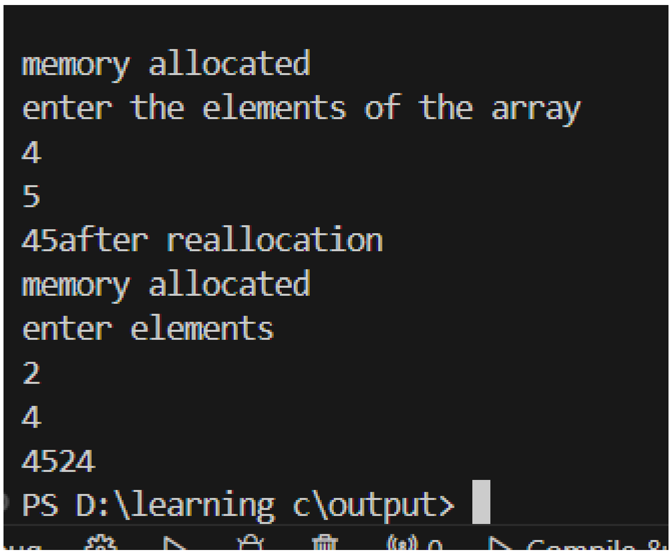
Objective: Write a program to dynamically allocate an integer array and allow the user to resize it.

Description:

1. The program should ask the user to enter the initial size of the array.
2. Allocate memory using malloc.
3. Allow the user to enter elements into the array.
4. Provide an option to increase or decrease the size of the array. Use realloc to adjust the size.
5. Print the elements of the array after each resizing operation.

```
//dynamic array resizing
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int *ptr;
   int num,i;

   printf("enter number of elements");
   scanf("%d",&num);
   printf("\n");
   ptr=(int *)malloc(num*sizeof(int));
   if(ptr==NULL)
   {
      printf("memory not allocated \n");
      exit(0);
   }
   else
   {
      printf("memory allocated \n");
   }
   printf("enter the elements of the array \n");
   for(i=0;i<num;i++)
   {
      scanf("%d",&ptr[i]);


   }
   for(i=0;i<num;i++)
   {
      printf("%d",ptr[i]);
   }
```

```
  // free(ptr);
  ptr=(int*)realloc(ptr,2*sizeof(int));
  printf("after reallocation \n");
  if(ptr==NULL)
   {
      printf("memory not allocated \n");
      exit(0);
   }
   else
   {
      printf("memory allocated \n");
   }
   printf("enter elements \n");
   for(i=num;i < num + 2;i++)
   {
      scanf("%d",&ptr[i]);

   }
   for(i=0;i<num+2;i++)
   {
      printf("%d",ptr[i]);
   }
   free(ptr);




}
```



```
memory allocated
enter the elements of the array
4
5
45after reallocation
memory allocated
enter elements
2
4
4524
PS D:\learning c\output>
```

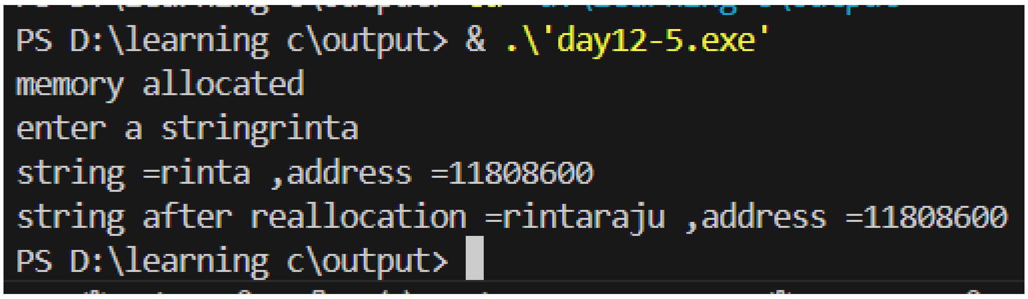
Problem 2: String Concatenation Using Dynamic Memory

Objective: Create a program that concatenates two strings using dynamic memory allocation.

Description:

1. Accept two strings from the user.
2. Use malloc to allocate memory for the first string.
3. Use realloc to resize the memory to accommodate the concatenated string.
4. Concatenate the strings and print the result.
5. Free the allocated memory.

```
//string concatenation using dynamic memory
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
   char *str;
   char str1[5]="raju";
   str=(char*)malloc(15);
   //strcpy(str,"jason");
   if(str==NULL)
   {
      printf("memory not allocated \n");
      exit(0);
   }
   else
   {
      printf("memory allocated \n");
   }
   printf("enter a string");
   scanf("%s",str);
   printf("string =%s ,address =%u \n",str,str);
   //reallocating memory(extending)
   str=(char*)realloc(str,25);
   strcat(str,str1);
   printf("string after reallocation =%s ,address =%u \n",str,str);
   free(str);
   return 0;

}
```



```
PS D:\learning c\output> & .\'day12-5.exe'
memory allocated
enter a stringrinta
string =rinta ,address =11808600
string after reallocation =rintaraju ,address =11808600
PS D:\learning c\output>
```

Problem 3: Sparse Matrix Representation

Objective: Represent a sparse matrix using dynamic memory allocation.

Description:

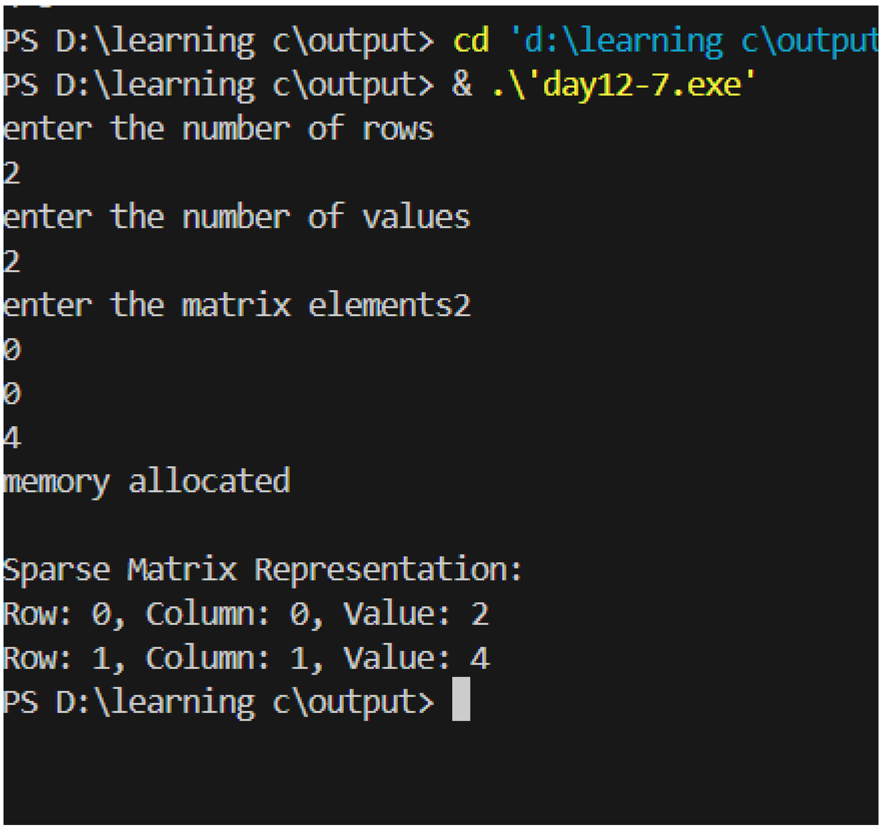
1. Accept a matrix of size m×nm \times nm×n from the user.
2. Store only the non-zero elements in a dynamically allocated array of structures (with fields for row, column, and value).

3. Print the sparse matrix representation.
4. Free the allocated memory at the end.

```
 //sparse matrix representation
#include<stdio.h>
struct sparseMatrix
{
   int row;
   int col;
   int value;


};
int main()
{
   struct sparseMatrix *spm;



   int r,c;
   int nonzero=0;
   printf("enter the number of rows \n");
   scanf("%d",&r);
   printf("enter the number of values\n");
   scanf("%d",&c);
   int sp[r][c];
   //getting elements from user
   printf("enter the matrix elements");
   for(int i=0;i<r;i++)
   {
      for(int j=0;j<c;j++)
      {
         scanf("%d",&sp[i][j]);
         if(sp[i][j]!=0)
```

```
        {
            nonzero++;
        }
    }
}
spm=(struct sparseMatrix*)malloc(nonzero*sizeof(struct sparseMatrix));
if(spm==NULL)
{
  printf("memory not allocated \n");

}
else
{
    printf("memory allocated \n");
}
int k = 0;
 for (int i = 0; i < r; i++)
 {
    for (int j = 0; j < c; j++)
    {
        if (sp[i][j] != 0)
        {
             spm[k].row = i;
             spm[k].col = j;
             spm[k].value = sp[i][j];
             k++;
             }
    }
  }
 printf("\nSparse Matrix Representation:\n");
 for (int i = 0; i < nonzero; i++)
```

```
    {
      printf("Row: %d, Column: %d, Value: %d\n", spm[i].row, spm[i].col, spm[i].value);

     }
   free(spm);
   return 0;

}
```



```
PS D:\learning c\output> cd 'd:\learning c\output
PS D:\learning c\output> & .\'day12-7.exe'
enter the number of rows
2
enter the number of values
2
enter the matrix elements2
0
0
4
memory allocated

Sparse Matrix Representation:
Row: 0, Column: 0, Value: 2
Row: 1, Column: 1, Value: 4
PS D:\learning c\output>
```

.

Problem 5: Dynamic 2D Array Allocation

Objective: Write a program to dynamically allocate a 2D array.

Description:

1. Accept the number of rows and columns from the user.
2. Use malloc (or calloc) to allocate memory for the rows and columns dynamically.
3. Allow the user to input values into the 2D array.
4. Print the array in matrix format.
5. Free all allocated memory at the end.

```
//dynamic 2D array allocation
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int *ptr;
  int rows,cols,i,j;

  printf("enter number of rows \n");
  scanf("%d",&rows);
  printf("enter number of column \n");
  scanf("%d",&cols);

  printf("\n");
  ptr=(int *)malloc(rows*cols*sizeof(int));
  if(ptr==NULL)
  {
    printf("memory not allocated \n");
    exit(0);
  }
  else
  {
    printf("memory allocated \n");
  }
  printf("enter the elements of the array \n");
  for (i = 0; i < rows; i++)
```

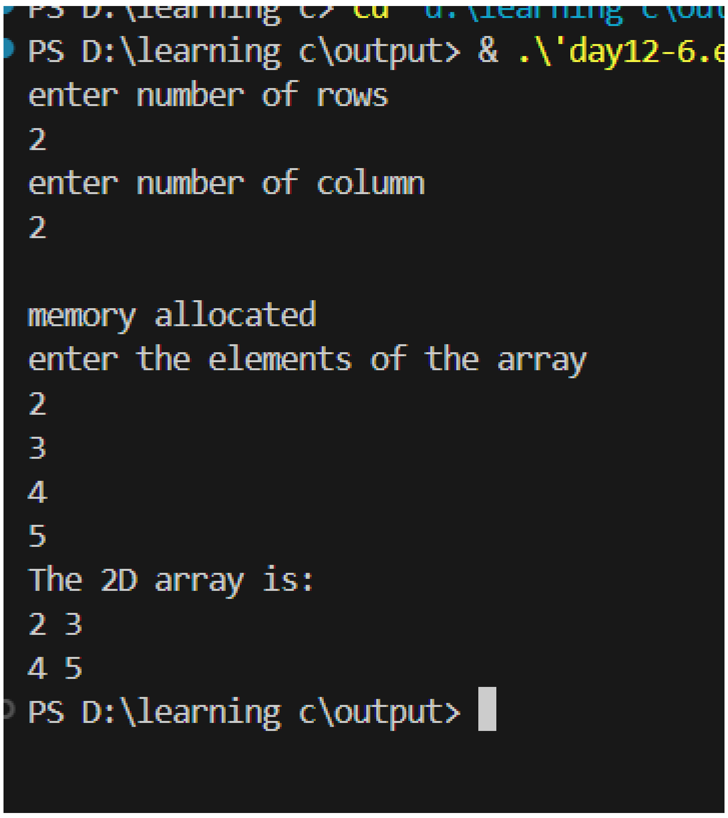
```
    {
        for (j = 0; j < cols; j++)
        {
            scanf("%d", &ptr[i * cols + j]);
            }
    }
    printf("The 2D array is:\n");
    for (i = 0; i < rows; i++)
    {
        for (j = 0; j < cols; j++)
        {
            printf("%d ", ptr[i * cols + j]);
        }
             printf("\n");
    }
   free(ptr);
}
```

/*To access an element in this 1D array as if it were a 2D array, you need to calculate its position using the formula i * cols + j.

Here, i is the row index and j is the column index.

i * cols gives you the start of the i-th row, and adding j moves you to the j-th column within that row.

This effectively converts the 2D index [i][j] into a 1D index i * cols + j.*/



```
PS D:\learning c> cd 'd:\learning c\out
PS D:\learning c\output> & .\'day12-6.e
enter number of rows
2
enter number of column
2

memory allocated
enter the elements of the array
2
3
4
5
The 2D array is:
2 3
4 5
PS D:\learning c\output> 
```

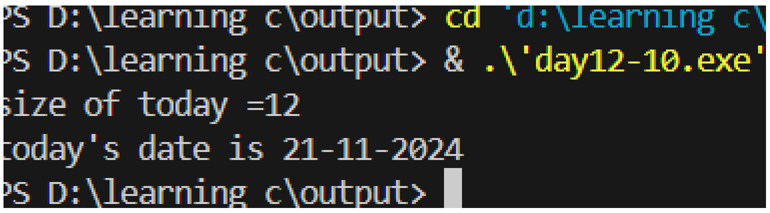
9. //structure

```
#include<stdio.h>
struct date
{
   int day;
   int month;
   int year;
}today;//can declare and define like this also
int main()
{
  // struct date today;//memory allocated
   printf("size of today =%d \n",sizeof(today));
   //accessing members in struct
   today.day=21;
   today.month=11;
   today.year=2024;
```

```
    printf("today's date is %d-%d-%d",today.day,today.month,today.year);
    return 0;
}
```

10. //un named structures

```
//structure
#include<stdio.h>
struct
{
    int day;
    int month;
    int year;
}today;
int main()
{

    printf("size of today =%d \n",sizeof(today));

    today.day=21;
    today.month=11;
    today.year=2024;
    printf("today's date is %d-%d-%d",today.day,today.month,today.year);
    return 0;
}
```

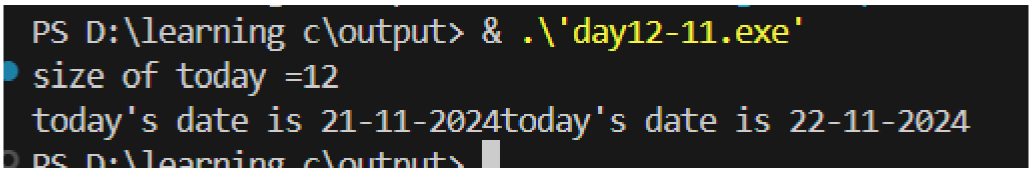


```
PS D:\learning c\output> cd 'd:\learning c\
PS D:\learning c\output> & .\'day12-10.exe'
ize of today =12
oday's date is 21-11-2024
PS D:\learning c\output>
```

11. //un named structures

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

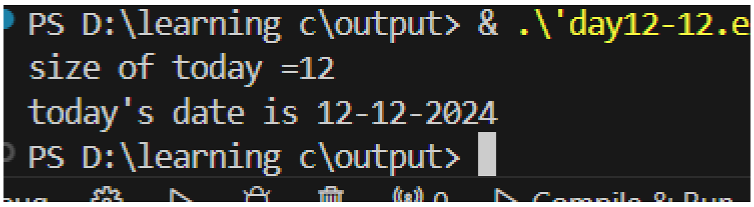
```c
struct
{
    int day;
    int month;
    int year;
}today,tomo;
int main()
{

    printf("size of today =%d \n",sizeof(today));

    today.day=21;
    today.month=11;
    today.year=2024;
    tomo.day=22;
    tomo.month=11;
    tomo.year=2024;
    printf("today's date is %d-%d-%d",today.day,today.month,today.year);
    printf("today's date is %d-%d-%d",tomo.day,tomo.month,tomo.year);

    return 0;
}
```



```
PS D:\learning c\output> & .\'day12-11.exe'
size of today =12
today's date is 21-11-2024today's date is 22-11-2024
PS D:\learning c\output>
```

12.

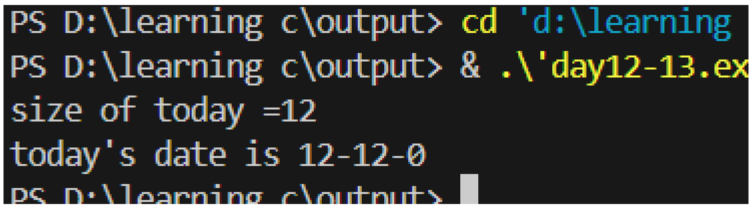
```c
#include<stdio.h>
struct date
{
    int day;
    int month;
    int year;
};
int main()
{
   struct date today={12,12,2024};//need tagname for this
    printf("size of today =%d \n",sizeof(today));



    printf("today's date is %d-%d-%d",today.day,today.month,today.year);


    return 0;
}
```



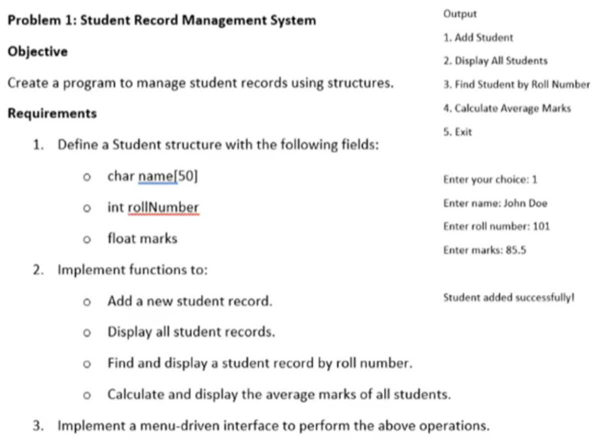
```
PS D:\learning c\output> & .\'day12-12.e
size of today =12
today's date is 12-12-2024
PS D:\learning c\output>
```

13.

```
#include<stdio.h>
struct date
{
    int day;
    int month;
    int year;
};
int main()
{
   struct date today={12,12};
    printf("size of today =%d \n",sizeof(today));



    printf("today's date is %d-%d-%d",today.day,today.month,today.year);
    return 0;

}
```



```
PS D:\learning c\output> cd 'd:\learning
PS D:\learning c\output> & .\'day12-13.ex
size of today =12
today's date is 12-12-0
PS D:\learning c\output>
```



Problem 1: Student Record Management System

Objective

Create a program to manage student records using structures.

Requirements

1. Define a Student structure with the following fields:
   - char name[50]
   - int rollNumber
   - float marks
2. Implement functions to:
   - Add a new student record.
   - Display all student records.
   - Find and display a student record by roll number.
   - Calculate and display the average marks of all students.
3. Implement a menu-driven interface to perform the above operations.

```
Output
1. Add Student
2. Display All Students
3. Find Student by Roll Number
4. Calculate Average Marks
5. Exit

Enter your choice: 1
Enter name: John Doe
Enter roll number: 101
Enter marks: 85.5

Student added successfully!
```

14.

```
#include<stdio.h>

struct student{

    char name[50];

    int rollnumber;

    float marks;

};

int main()

{

    int max=50;

    struct student std[max];

    int choice;

    int count=0;

    int roll;
```

```
while(1)
{
   printf("1.Add Student\n");
   printf("2.Displace all students \n");
   printf("3.find student by roll number \n");
   printf("4.calculate average marks \n");
   printf("5.exit\n");
   printf("enter a choice \n");
   scanf("%d",&choice);
   switch(choice)
   {
      case 1:
         if(count>=max)
         {
            printf("no more student can be added \n");
         }
         printf("enter a name");
         scanf("%s",std[count].name);
         printf("enter roll number");
         scanf("%d",&std[count].rollnumber);
         printf("enter marks :");
         scanf("%f",&std[count].marks);
         count++;
         printf("student added sucessfully");
         break;
      case 2:
         if(count==0)
         {
            printf("no student records available \n");
         }
         else
```

```
            {
                for(int i=0;i<count;i++)
                {
                    printf("Name: %s, Roll Number: %d, Marks: %.2f\n", std[i].name, std[i].rollnumber,
std[i].marks);
                }


            }
            break;
        case 3:


            printf("enter a roll number");
            scanf("%d",&roll);
            int found=0;
            for(int i=0;i<count;i++)
            {
                if(std[i].rollnumber==roll)
                {
                    printf("name:%s , rollnumber:%d , marks :%f
\n",std[i].name,std[i].rollnumber,std[i].marks);
                    found=1;
                    break;
                }


            }
            if(!found)
            {
                printf("sorry student not found");
            }
            break;
        case 4:
            if(count==0)
            {
                printf("no student records");
            }
            else
            {
                float totalmarks=0;
                for(int i=0;i<count;i++)
                {
                    totalmarks+=std[i].marks;
                }
                float avg=totalmarks/count;
                printf("average marks:%f \n",avg);
            }
            break;
        case 5:
                return 0;
        default:
            break;




    }


  }


}
```

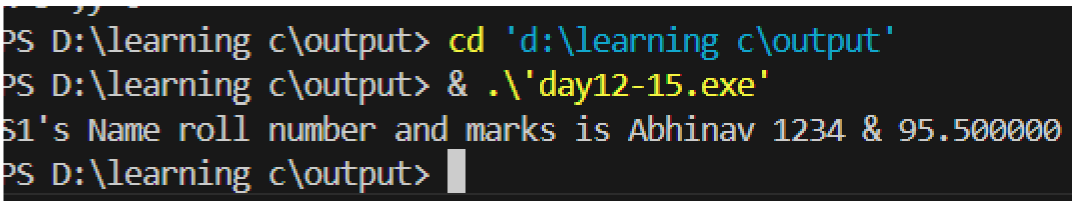
```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day12-14.exe'
1.Add Student
2.Displace all students
```

```
3.find student by roll number

4.calculate average marks

5.exit

enter a choice

1

enter a namerinta

enter roll number34

enter marks :22

student added sucessfully1.Add Student

2.Displace all students

3.find student by roll number

4.calculate average marks

5.exit

enter a choice

2

Name: rinta, Roll Number: 34, Marks: 22.00

1.Add Student

2.Displace all students

3.find student by roll number

4.calculate average marks

5.exit

enter a choice

3

enter a roll number1

sorry student not found1.Add Student

2.Displace all students

3.find student by roll number

4.calculate average marks

5.exit

enter a choice

3
```

```
enter a roll number34
name:rinta , rollnumber:34 , marks :22.000000
1.Add Student
2.Displace all students
3.find student by roll number
4.calculate average marks
5.exit
enter a choice
4
average marks:22.000000
1.Add Student
2.Displace all students
3.find student by roll number
4.calculate average marks
5.exit
enter a choice
5
PS D:\learning c\output>
```

15. #include <stdio.h>

```c
struct student{

    char name[50];

    int rollNumber;

    float marks;

};

int main(){
    //struct date today;

    struct student s1 = {.rollNumber = 1234, .name = "Abhinav", .marks = 95.5};

    printf("S1's Name roll number and marks is %s %d & %f \n", s1.name, s1.rollNumber,s1.marks);

    return 0;

}
```

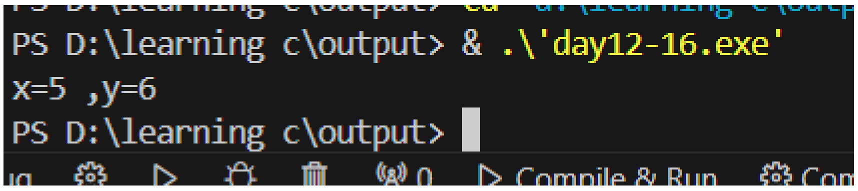


```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day12-15.exe'
S1's Name roll number and marks is Abhinav 1234 & 95.500000
PS D:\learning c\output>
```

16. //assignment with compound literals

```
#include<stdio.h>

struct coordinate{

    int x;

    int y;

};

void printcoordinate(struct coordinate);

int main()

{    printcoordinate((struct coordinate){5,6});//compound literal

    //struct coordinate pointA={2,3};

    //printcoordinate(pointA);

    return 0;

}

void printcoordinate(struct coordinate temp)

{

    printf("x=%d ,y=%d",temp.x,temp.y);
```

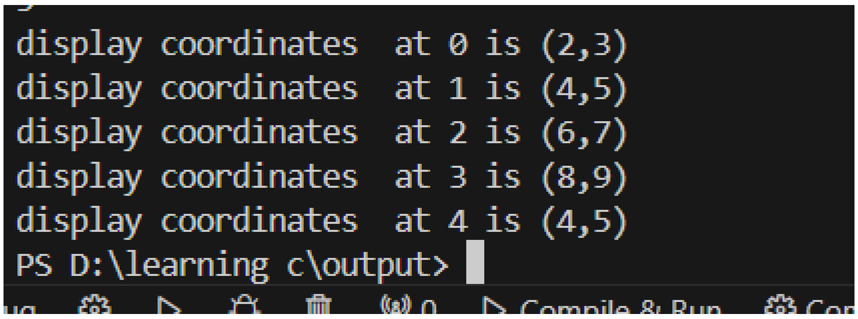
}



```
PS D:\learning c\output> & .\'day12-16.exe'
x=5 ,y=6
PS D:\learning c\output>
```

17. //array of structures

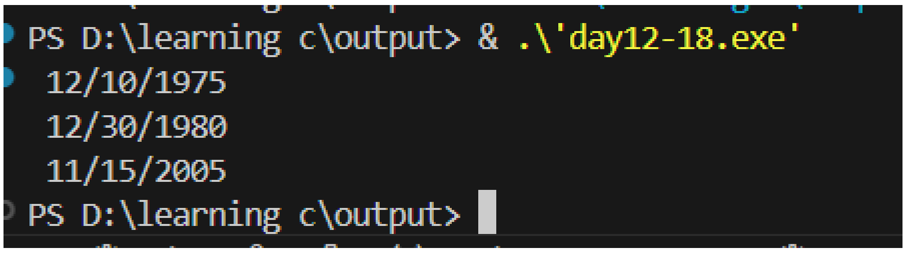
```
#include<stdio.h>
struct coordinate{
   int x;
   int y;
};
int main()
{   struct coordinate pnt[5];
   //pnt[0].x=1;
  // pnt[0].y=7;
   for(int i=0;i<5;i++)
   {
       printf("intialize the struct present in index %d\n ",i);
       scanf("%d %d",&pnt[i].x,&pnt[i].y);
   }
   //pnt[0].y=7;
   for(int i=0;i<5;i++)
   {
       printf("display coordinates  at %d is (%d,%d)\n",i,pnt[i].x,pnt[i].y);
   }

   return 0;
}
```



```
display coordinates  at 0 is (2,3)
display coordinates  at 1 is (4,5)
display coordinates  at 2 is (6,7)
display coordinates  at 3 is (8,9)
display coordinates  at 4 is (4,5)
PS D:\learning c\output>
```

18.

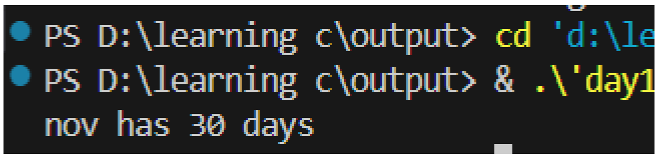
```
#include<stdio.h>

struct date{

  int day;

  int month;

  int year;

};

int main()

{

   struct date mydates[5]={{12,10,1975},{12,30,1980},{11,15,2005}};

   for(int i=0;i<3;i++)

   {

      printf(" %d/%d/%d \n",mydates[i].day,mydates[i].month,mydates[i].year);


   }

}
```



```
PS D:\learning c\output> & .\'day12-18.exe'
 12/10/1975
 12/30/1980
 11/15/2005
PS D:\learning c\output>
```

19.

```
#include<stdio.h>

struct month

{

   int noofdays;

   char name[5];
};

int main()

{

struct month testmonth= {.noofdays=30,.name="nov"};


    printf("%s has %d days \n",testmonth.name,testmonth.noofdays);


}
```



```
PS D:\learning c\output> cd 'd:\le
PS D:\learning c\output> & .\'day1
nov has 30 days
```

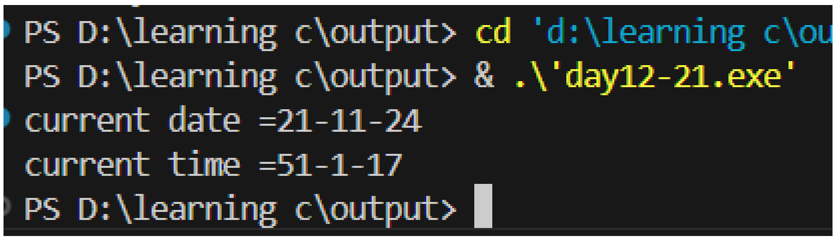
20.

```
#include <stdio.h>


struct Month{


    int noOfDays;


    char name[3];


};


int main(){


    struct Month allMonths[12];


    for(int i = 0; i < 12; i++){


        printf("Enter The Month Name and the no. of days associated with that month");


        scanf("%s  %d", allMonths[i].name, &allMonths[i].noOfDays);
        printf("\n");

    }

    for(int j = 0; j < 12; j++){

        printf("Name of the Month = %s having %d \n",allMonths[j].name,allMonths[j].noOfDays);

    }
}
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day12-20.exe'
Enter The Month Name and the no. of days associated with that month jan 30


Enter The Month Name and the no. of days associated with that monthfeb 30


Enter The Month Name and the no. of days associated with that monthmarch 22


Enter The Month Name and the no. of days associated with that monthapril 30


Enter The Month Name and the no. of days associated with that monthmay 31


Enter The Month Name and the no. of days associated with that month june
30


Enter The Month Name and the no. of days associated with that monthjuly 31


Enter The Month Name and the no. of days associated with that monthaug 31


Enter The Month Name and the no. of days associated with that monthsept 30
Enter The Month Name and the no. of days associated with that monthoct 31

Enter The Month Name and the no. of days associated with that monthnov 30

Enter The Month Name and the no. of days associated with that monthdec 31

Name of the Month = jan having 30
Name of the Month = feb having 30
Name of the Month = marc▲ having 22
Name of the Month = apri▼ having 30
Name of the Month = may having 31
Name of the Month = june▼ having 30
Name of the Month = july▼ having 31
Name of the Month = aug having 31
Name of the Month = sept▼ having 30
Name of the Month = oct having 31
Name of the Month = nov having 30
Name of the Month = dec having 31
PS D:\learning c\output>
```

21. //nested structures

```
#include <stdio.h>

struct currentDate{

    int day;
    int month;
    int year;

};
```

```
struct currentTime{
    int sec;
    int min;
    int hrs;
};
struct cDateTime{
    struct currentDate d1;
    struct currentTime t1;
} ;
int main()
{
    struct cDateTime dt={{21,11,24},{51,01,17}};


    printf("current date =%d-%d-%d\n",dt.d1.day,dt.d1.month,dt.d1.year);
    printf("current time =%d-%d-%d\n",dt.t1.sec,dt.t1.min,dt.t1.hrs);



    return 0;
}
```



```
PS D:\learning c\output> cd 'd:\learning c\ou
PS D:\learning c\output> & .\'day12-21.exe'
current date =21-11-24
current time =51-1-17
PS D:\learning c\output>
```

22. //struct inside struct

```
//nested structures


#include <stdio.h>



struct currentTime{
   struct currentDate
```

```
  {


     int day;

     int month;

     int year;


   };

   int sec;

   int min;

   int hrs;

};

struct cDateTime{

   struct currentDate d1;

   struct currentTime t1;

} ;

int main()

{

   struct cDateTime dt={{21,11,24},{51,01,17}};


   printf("current date =%d-%d-%d\n",dt.d1.day,dt.d1.month,dt.d1.year);

   printf("current time =%d-%d-%d\n",dt.t1.sec,dt.t1.min,dt.t1.hrs);



   return 0;

}
```

23.Problem 1: Employee Management System

Objective: Create a program to manage employee details using structures.

Description:

1. Define a structure Employee with fields:

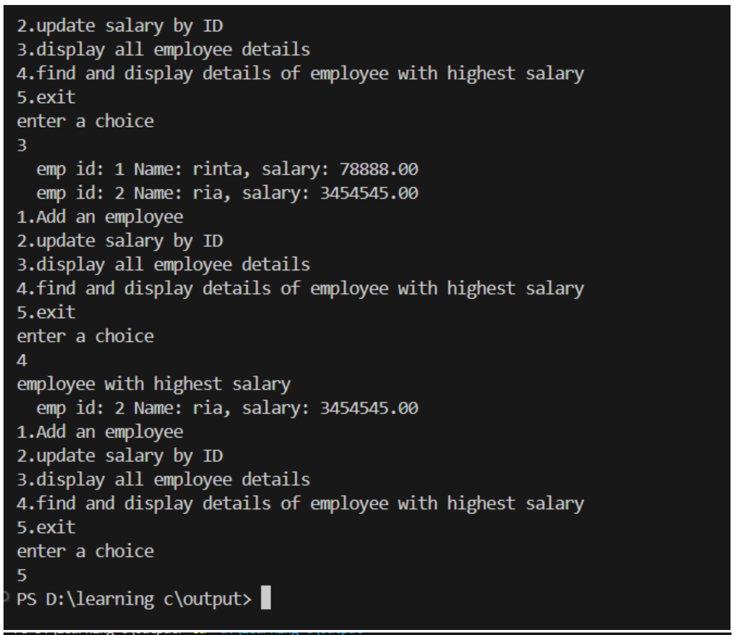
   - int emp_id: Employee ID
   - char name[50]: Employee name
   - float salary: Employee salary
2. Write a menu-driven program to:
   - Add an employee.
   - Update employee salary by ID.
   - Display all employee details.
   - Find and display details of the employee with the highest salary.

```
//employee management system
#include<stdio.h>
struct Employee{
  int empid;
  char name[50];
  float salary;
};
int main()
{
  int max=50;
  struct Employee emp[max];
  int choice;
  int count=0;
  int empid;

  while(1)
  {
    printf("1.Add an employee\n");
    printf("2.update salary by ID \n");
    printf("3.display all employee details\n");
    printf("4.find and display details of employee with highest salary \n");
    printf("5.exit\n");
    printf("enter a choice \n");
```

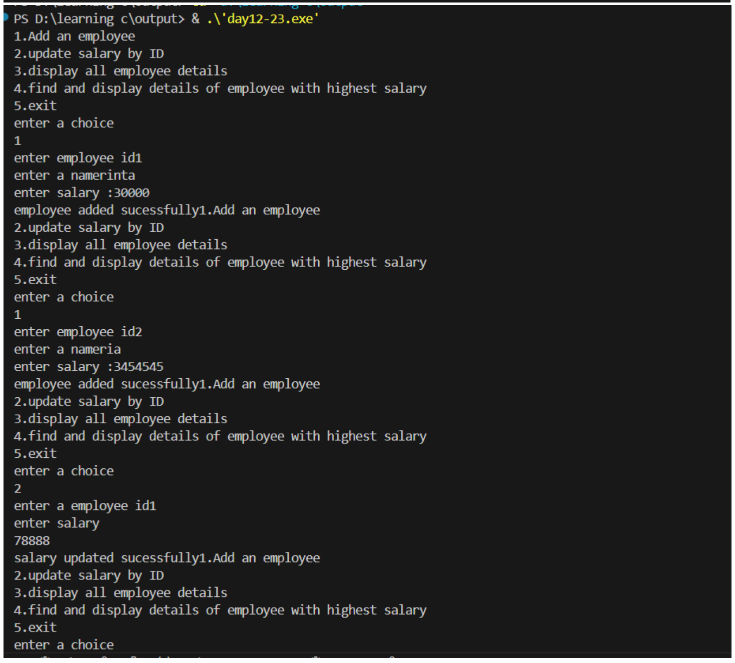
```
scanf("%d",&choice);
switch(choice)
{
   case 1:
      if(count>=max)
      {
         printf("no more employee can be added \n");
      }
      printf("enter employee id");
      scanf("%d",&emp[count].empid);
      printf("enter a name");
      scanf("%s",emp[count].name);
      printf("enter salary :");
      scanf("%f",&emp[count].salary);
      count++;
      printf("employee added sucessfully");
      break;
   case 2:

      printf("enter a employee id");
      scanf("%d",&empid);
      int found=0;
      for(int i=0;i<count;i++)
      {
         if(emp[i].empid==empid)
         {
            printf("enter salary \n");
            scanf("%f",&emp[i].salary);

            found=1;
            printf("salary updated sucessfully");
```

```
                break;


              }


            }

            if(!found)

            {

              printf("sorry employee not found");

            }

            break;

        case 3:

            if(count==0)

            {

              printf("no student records available \n");

            }

            else

            {

              for(int i=0;i<count;i++)

              {

                printf("  emp id: %d Name: %s, salary: %.2f\n",
emp[i].empid,emp[i].name,emp[i].salary);

              }


            }

            break;


        case 4:

            if(count==0)

            {

              printf("no employee records");

            }
            else
            {
                int index=0;
                for(int i=0;i<count;i++)
                {
                    if(emp[i].salary>=emp[index].salary)
                    {
                        index=i;


                    }


                }
                printf("employee with highest salary \n");
                printf("  emp id: %d Name: %s, salary: %.2f\n",
emp[index].empid,emp[index].name,emp[index].salary);




            }
            break;
        case 5:
                return 0;
        default:
            break;





    }
```

```
        }


    }
```



```
2.update salary by ID
3.display all employee details
4.find and display details of employee with highest salary
5.exit
enter a choice
3
  emp id: 1 Name: rinta, salary: 78888.00
  emp id: 2 Name: ria, salary: 3454545.00
1.Add an employee
2.update salary by ID
3.display all employee details
4.find and display details of employee with highest salary
5.exit
enter a choice
4
employee with highest salary
  emp id: 2 Name: ria, salary: 3454545.00
1.Add an employee
2.update salary by ID
3.display all employee details
4.find and display details of employee with highest salary
5.exit
enter a choice
5
PS D:\learning c\output>
```



```
PS D:\learning c\output> & .\'day12-23.exe'
1.Add an employee
2.update salary by ID
3.display all employee details
4.find and display details of employee with highest salary
5.exit
enter a choice
1
enter employee id1
enter a namerinta
enter salary :30000
employee added sucessfully1.Add an employee
2.update salary by ID
3.display all employee details
4.find and display details of employee with highest salary
5.exit
enter a choice
1
enter employee id2
enter a nameria
enter salary :3454545
employee added sucessfully1.Add an employee
2.update salary by ID
3.display all employee details
4.find and display details of employee with highest salary
5.exit
enter a choice
2
enter a employee id1
enter salary
78888
salary updated sucessfully1.Add an employee
2.update salary by ID
3.display all employee details
4.find and display details of employee with highest salary
5.exit
enter a choice
```

Problem 2: Library Management System

Objective: Manage a library system with a structure to store book details.

Description:

1. Define a structure Book with fields:
   - int book_id: Book ID
   - char title[100]: Book title
   - char author[50]: Author name
   - int copies: Number of available copies
2. Write a program to:
   - Add books to the library.
   - Issue a book by reducing the number of copies.
   - Return a book by increasing the number of copies.
   - Search for a book by title or author name.

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

struct Book {
   int bookid;
   char title[100];
   char author[50];
   int copies;
};

int main() {
   int max = 50;
   struct Book book[max];
   int choice;
   int count = 0;
   char searchTitle[100], searchAuthor[50];
```

```
while (1) {
  printf("\n1. Add books to library\n");
  printf("2. Issue a book by reducing the number of copies\n");
  printf("3. Return a book by increasing the number of copies\n");
  printf("4. Search for a book by title or author name\n");
  printf("5. Exit\n");
  printf("Enter your choice: ");
  scanf("%d", &choice);

  switch (choice) {
    case 1:
      if (count >= max) {
        printf("No more books can be added.\n");
      } else {
        printf("Enter book ID: ");
        scanf("%d", &book[count].bookid);
        printf("Enter book title: ");
        scanf(" %s", book[count].title);
        printf("Enter author name: ");
        scanf(" %s", book[count].author);
        printf("Enter the number of copies: ");
        scanf("%d", &book[count].copies);
        count++;
        printf("Book added successfully.\n");
      }
      break;

    case 2:
      printf("Enter book ID to issue: ");
      int issueBookID;
      scanf("%d", &issueBookID);
    int found = 0;
    for (int i = 0; i < count; i++) {
        if (book[i].bookid == issueBookID) {
            if (book[i].copies > 0) {
                book[i].copies--;
                printf("Book issued successfully.\n");
            } else {
                printf("No copies available.\n");
            }
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Book not found.\n");
    }
    break;

case 3:
    printf("Enter book ID to return: ");
    int returnBookID;
    scanf("%d", &returnBookID);
    found = 0;
    for (int i = 0; i < count; i++) {
        if (book[i].bookid == returnBookID) {
            book[i].copies++;
            printf("Book returned successfully.\n");
            found = 1;
            break;
        }
    }
```

```
                if (!found) {
                    printf("Book not found.\n");
                }
                break;

            case 4:
                printf("Enter book title or author name to search: ");
                scanf(" %s", searchTitle);
                found = 0;
                for (int i = 0; i < count; i++) {
                    if (strstr(book[i].title, searchTitle) != NULL || strstr(book[i].author, searchTitle) != NULL) {
                        printf("Book ID: %d, Title: %s, Author: %s, Copies: %d\n", book[i].bookid, book[i].title, book[i].author, book[i].copies);
                        found = 1;
                    }
                }
                if (!found) {
                    printf("Book not found.\n");
                }
                break;

            case 5:
                return 0;

            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```

```
 1. Add books to library
2. Issue a book by reducing the number of copies
3. Return a book by increasing the number of copies
4. Search for a book by title or author name
5. Exit
Enter your choice: 1
Enter book ID: 1
Enter book title: famous
Enter author name: rexa
Enter the number of copies: 4
Book added successfully.

1. Add books to library
2. Issue a book by reducing the number of copies
3. Return a book by increasing the number of copies
4. Search for a book by title or author name
5. Exit
Enter your choice: 1
Enter book ID: 2
Enter book title: yuma
Enter author name: rio
Enter the number of copies: 4
Book added successfully.

1. Add books to library
2. Issue a book by reducing the number of copies
3. Return a book by increasing the number of copies
4. Search for a book by title or author name
5. Exit
```

```
Enter your choice: 2
Enter book ID to issue: 1
Book issued successfully.

1. Add books to library
2. Issue a book by reducing the number of copies
3. Return a book by increasing the number of copies
4. Search for a book by title or author name
5. Exit
Enter your choice: 3
Enter book ID to return: 1
Book returned successfully.

1. Add books to library
2. Issue a book by reducing the number of copies
3. Return a book by increasing the number of copies
4. Search for a book by title or author name
5. Exit
Enter your choice: 4
Enter book title or author name to search: famous
Book ID: 1, Title: famous, Author: rexa, Copies: 4

1. Add books to library
2. Issue a book by reducing the number of copies
3. Return a book by increasing the number of copies
4. Search for a book by title or author name
5. Exit
Enter your choice: 5
PS D:\learning c\output>
```

Problem 3: Cricket Player Statistics

Objective: Store and analyze cricket player performance data.

Description:

1. Define a structure Player with fields:
   - char name[50]: Player name
   - int matches: Number of matches played
   - int runs: Total runs scored
   - float average: Batting average
2. Write a program to:
   - Input details for n players.
   - Calculate and display the batting average for each player.
   - Find and display the player with the highest batting average.

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

struct Player {
   char name[50];
   int matches;
   int runs;
   float average;
};

int main() {
   int max = 50;
   struct Player players[max];
   int count = 0;
   int choice;

   while (1) {
```

```
printf("\n1. Add player details\n");
printf("2. Display all player statistics\n");
printf("3. Find and display the player with the highest batting average\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
  case 1:
    if (count >= max) {
      printf("Cannot add more players. Maximum limit reached.\n");
    } else {
      printf("Enter name: ");
      scanf(" %s", players[count].name);
      printf("Number of matches: ");
      scanf("%d", &players[count].matches);
      printf("Total runs scored: ");
      scanf("%d", &players[count].runs);



      if (players[count].matches != 0) {
        players[count].average = (float)players[count].runs / players[count].matches;
      } else {
        players[count].average = 0.0;
      }

      count++;
      printf("Player details added successfully.\n");
    }
    break;
```

```
        case 2:
            if (count == 0) {
                printf("No player records available.\n");
            } else {
                printf("\nPlayer Statistics:\n");
                for (int i = 0; i < count; i++) {
                    printf("Name: %s, Matches: %d, Runs: %d, Average: %.2f\n", players[i].name, players[i].matches, players[i].runs, players[i].average);
                }
            }
            break;

        case 3:
            if (count == 0) {
                printf("No player records available.\n");
            } else {
                int highestIndex = 0;
                for (int i = 1; i < count; i++) {
                    if (players[i].average > players[highestIndex].average) {
                        highestIndex = i;
                    }
                }
                printf("\nPlayer with the highest batting average:\n");
                printf("Name: %s, Matches: %d, Runs: %d, Average: %.2f\n", players[highestIndex].name, players[highestIndex].matches, players[highestIndex].runs, players[highestIndex].average);
            }
            break;

        case 4:
            return 0;
            default:
                printf("Invalid choice! Please try again.\n");
                break;
        }
    }

    return 0;
}
```

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day12-25.exe'

1. Add player details
2. Display all player statistics
3. Find and display the player with the highest batting average
4. Exit
Enter your choice: 1
Enter name: tom
Number of matches: 200
Total runs scored: 12000
Player details added successfully.

1. Add player details
2. Display all player statistics
3. Find and display the player with the highest batting average
4. Exit
Enter your choice: 1
Enter name: yin
Number of matches: 100
Total runs scored: 800
Player details added successfully.
```

```
1. Add player details
2. Display all player statistics
3. Find and display the player with the highest batting average
4. Exit
Enter your choice: 2

Player Statistics:
Name: tom, Matches: 200, Runs: 12000, Average: 60.00
Name: yin, Matches: 100, Runs: 800, Average: 8.00

1. Add player details
2. Display all player statistics
3. Find and display the player with the highest batting average
4. Exit
Enter your choice: 3

Player with the highest batting average:
Name: tom, Matches: 200, Runs: 12000, Average: 60.00

1. Add player details
2. Display all player statistics
3. Find and display the player with the highest batting average
4. Exit
Enter your choice: 4
PS D:\learning c\output>
```

26.

Problem 4: Student Grading System

Objective: Manage student data and calculate grades based on marks.

Description:

1. Define a structure Student with fields:
   - int roll_no: Roll number
   - char name[50]: Student name
   - float marks[5]: Marks in 5 subjects
   - char grade: Grade based on the average marks
2. Write a program to:
   - Input details of n students.
   - Calculate the average marks and assign grades (A, B, C, etc.).
   - Display details of students along with their grades.

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

struct Student {
   int roll_no;
   char name[50];
   float marks[5];
   char grade;
};

int main() {
   int max;
   printf("enter number of students \n");
   scanf("%d",&max);
   struct Student students[max];
   int count = 0;
   int choice;
   int roll_no;

   while (1) {
      printf("\n1. input details of students\n");
printf("2. calculate average, and assign grade\n");
printf("3. Display all student details  along with grades\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
  case 1:
    if (count >= max) {
      printf("Cannot add more students. Maximum limit reached.\n");
    } else {
      printf("Enter roll number: ");
      scanf("%d", &students[count].roll_no);
      printf("Enter name: ");
      scanf(" %s", students[count].name);
      printf("Roll number and name added successfully.\n");
      count++;
    }
    break;

  case 2:
    if (count == 0) {
      printf("No students available. Please add students first.\n");
    } else {
      printf("Enter roll number to input marks and calculate grade: ");
      scanf("%d", &roll_no);
      int found = 0;
      for (int i = 0; i < count; i++) {
        if (students[i].roll_no == roll_no) {
          float total_marks = 0;
          for (int j = 0; j < 5; j++) {
                printf("Enter marks for subject %d: ", j + 1);
                scanf("%f", &students[i].marks[j]);
                total_marks += students[i].marks[j];
            }


            float average = total_marks / 5.0;


            if (average >= 90) {
                students[i].grade = 'A';
            } else if (average >= 80) {
                students[i].grade = 'B';
            } else if (average >= 70) {
                students[i].grade = 'C';
            } else if (average >= 60) {
                students[i].grade = 'D';
            } else {
                students[i].grade = 'F';
            }

            printf("Average marks: %.2f, Grade: %c\n", average, students[i].grade);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Student with roll number %d not found.\n", roll_no);
    }
}
break;
```

```
        case 3:
            if (count == 0) {
                printf("No student records available.\n");
            } else {
                printf("\nStudent Details:\n");
                for (int i = 0; i < count; i++) {
                    printf("Roll No: %d, Name: %s, Marks: [%.2f, %.2f, %.2f, %.2f, %.2f], Grade:
%c\n",
                        students[i].roll_no, students[i].name, students[i].marks[0],
students[i].marks[1],
                        students[i].marks[2], students[i].marks[3], students[i].marks[4],
students[i].grade);
                }
            }
            break;

        case 4:
            return 0;

        default:
            printf("Invalid choice! Please try again.\n");
            break;
        }
    }

    return 0;
}
```

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day12-26.exe'
enter number of students
2

1. input details of students
2. calculate average, and assign grade
3. Display all student details  along with grades
4. Exit
Enter your choice: 1
Enter roll number: 1
Enter name: rinta
Roll number and name added successfully.

1. input details of students
2. calculate average, and assign grade
3. Display all student details  along with grades
4. Exit
Enter your choice: 2
Enter roll number to input marks and calculate grade: 1
Enter marks for subject 1: 34
Enter marks for subject 2: 56
Enter marks for subject 3: 78
Enter marks for subject 4: 90
Enter marks for subject 5: 23
Average marks: 56.20, Grade: F

1. input details of students
2. calculate average, and assign grade
3. Display all student details  along with grades
4. Exit
Enter your choice: 1
Enter roll number: 2

Enter name: rani

Roll number and name added successfully.


1. input details of students

2. calculate average, and assign grade

3. Display all student details  along with grades

4. Exit

Enter your choice: 2

Enter roll number to input marks and calculate grade: 2

Enter marks for subject 1: 56

Enter marks for subject 2: 78

Enter marks for subject 3: 90

Enter marks for subject 4: 90

Enter marks for subject 5: 21

Average marks: 67.00, Grade: D


1. input details of students

2. calculate average, and assign grade

3. Display all student details  along with grades

4. Exit

Enter your choice: 3


Student Details:

Roll No: 1, Name: rinta, Marks: [34.00, 56.00, 78.00, 90.00, 23.00], Grade: F

Roll No: 2, Name: rani, Marks: [56.00, 78.00, 90.00, 90.00, 21.00], Grade: D


1. input details of students

2. calculate average, and assign grade

3. Display all student details  along with grades

4. Exit
```

```
Enter your choice: 4
PS D:\learning c\output>
```

27.

Problem 5: Flight Reservation System

Objective: Simulate a simple flight reservation system using structures.

Description:

1. Define a structure Flight with fields:
   - char flight_number[10]: Flight number
   - char destination[50]: Destination city
   - int available_seats: Number of available seats
2. Write a program to:
   - Add flights to the system.
   - Book tickets for a flight, reducing available seats accordingly.
   - Display the flight details based on destination.
   - Cancel tickets, increasing the number of available seats.

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

struct Flight {
   char flight_number[10];
   char destination[50];
   int available_seats;
};

int main() {
   int max_flights = 50;
   struct Flight flights[max_flights];
   int count = 0;
int choice;


while (1) {
    printf("\n1. Add a flight\n");
    printf("2. Book tickets for a flight\n");
    printf("3. Display flight details by destination\n");
    printf("4. Cancel tickets for a flight\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);


    switch (choice) {
        case 1:
            if (count >= max_flights) {
                printf("Cannot add more flights. Maximum limit reached.\n");
            } else {
                printf("Enter flight number: ");
                scanf("%s", flights[count].flight_number);
                printf("Enter destination: ");
                scanf(" %[^\n]%*c", flights[count].destination); // Reading string with spaces
                printf("Enter number of available seats: ");
                scanf("%d", &flights[count].available_seats);
                count++;
                printf("Flight added successfully.\n");
            }
            break;


        case 2:
            if (count == 0) {
                printf("No flights available. Please add flights first.\n");
            } else {
                char flight_number[10];
                int tickets;
                printf("Enter flight number to book tickets: ");
                scanf("%s", flight_number);
                printf("Enter number of tickets to book: ");
                scanf("%d", &tickets);

                int found = 0;
                for (int i = 0; i < count; i++) {
                    if (strcmp(flights[i].flight_number, flight_number) == 0) {
                        if (flights[i].available_seats >= tickets) {
                            flights[i].available_seats -= tickets;
                            printf("Tickets booked successfully. Remaining seats: %d\n",
flights[i].available_seats);
                        } else {
                            printf("Not enough available seats. Remaining seats: %d\n",
flights[i].available_seats);
                        }
                        found = 1;
                        break;
                    }
                }
                if (!found) {
                    printf("Flight not found.\n");
                }
            }
            break;

        case 3:
            if (count == 0) {
                printf("No flights available. Please add flights first.\n");
            } else {
```

```
            char destination[50];
            printf("Enter destination to search for flights: ");
            scanf(" %[^\n]%*c", destination); // Reading string with spaces

            int found = 0;
            for (int i = 0; i < count; i++) {
                if (strcmp(flights[i].destination, destination) == 0) {
                    printf("Flight Number: %s, Destination: %s, Available Seats: %d\n",
flights[i].flight_number, flights[i].destination, flights[i].available_seats);
                    found = 1;
                }
            }
            if (!found) {
                printf("No flights found for the destination: %s\n", destination);
            }
        }
        break;

    case 4:
        if (count == 0) {
            printf("No flights available. Please add flights first.\n");
        } else {
            char flight_number[10];
            int tickets;
            printf("Enter flight number to cancel tickets: ");
            scanf("%s", flight_number);
            printf("Enter number of tickets to cancel: ");
            scanf("%d", &tickets);

            int found = 0;
            for (int i = 0; i < count; i++) {
                if (strcmp(flights[i].flight_number, flight_number) == 0) {
                    flights[i].available_seats += tickets;
                    printf("Tickets canceled successfully. Updated seats: %d\n", flights[i].available_seats);
                    found = 1;
                    break;
                }
            }
            if (!found) {
                printf("Flight not found.\n");
            }
        }
        break;

    case 5:
        return 0;

    default:
        printf("Invalid choice! Please try again.\n");
        break;
    }
  }

  return 0;
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day12-27.exe'

1. Add a flight
2. Book tickets for a flight
3. Display flight details by destination
```

```
4. Cancel tickets for a flight

5. Exit

Enter your choice: 1

Enter flight number: 67788

Enter destination: dubai

Enter number of available seats: 400

Flight added successfully.


1. Add a flight

2. Book tickets for a flight

3. Display flight details by destination

4. Cancel tickets for a flight

5. Exit

Enter your choice: 2

Enter flight number to book tickets: 67788

Enter number of tickets to book: 37

Tickets booked successfully. Remaining seats: 363


1. Add a flight

2. Book tickets for a flight

3. Display flight details by destination

4. Cancel tickets for a flight

5. Exit

Enter your choice: 3

Enter destination to search for flights: dubai

Flight Number: 67788, Destination: dubai, Available Seats: 363


1. Add a flight

2. Book tickets for a flight

3. Display flight details by destination

4. Cancel tickets for a flight
```

```
5. Exit

Enter your choice: 4

Enter flight number to cancel tickets: 67788

Enter number of tickets to cancel: 10

Tickets canceled successfully. Updated seats: 373


1. Add a flight

2. Book tickets for a flight

3. Display flight details by destination

4. Cancel tickets for a flight

5. Exit

Enter your choice: 3

Enter destination to search for flights: dubai

Flight Number: 67788, Destination: dubai, Available Seats: 373


1. Add a flight

2. Book tickets for a flight

3. Display flight details by destination

4. Cancel tickets for a flight

5. Exit

Enter your choice: 5

PS D:\learning c\output>
```