DAY_12_ASSIGNMENT

Problem 1: Employee Management System
Objective: Create a program to manage employee details using structures.
Description:
Define a structure Employee with fields:
int emp_id: Employee ID
char name[50]: Employee name
float salary: Employee salary
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.

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

struct Employee{
    int emp_id;
    char name[50];
    float salary;
};
void add(struct Employee emp[],int *count);
void updated(struct Employee emp[],int count);
void display(struct Employee emp[],int count);
void heighestsalary(struct Employee emp[],int count);


int main(){

    struct Employee emp[100];
    int count =0;
    int op;
    while(1){
    printf("enter 1 to add \n");
    printf("enter 2 to update emp salary\n");
    printf("enter 3 display allemp details\n");
    printf("enter 4 to display the heighest salary\n");
    printf("enter your choice\n");
    scanf("%d",&op);
```

```
        switch (op)
        {
        case 1:
        add(emp,&count);
        break;

        case 2:
        updated(emp,count);
        break;

        case 3:
        display(emp,count);
        break;

        case 4:
        heighestsalary(emp,count);
        break;

        default:
            break;
        }
    }
}
void add(struct Employee emp[],int *count){
    printf("enter the name");
    scanf("%s",emp[*count].name);
    printf("enter the employe id");
    scanf("%d",&emp[*count].emp_id);
    printf("enter the current salary");
    scanf("%f",&emp[*count].salary);
    (*count)++;
    printf("added successfully");
}
void updated(struct Employee emp[],int count){
    int id,fount =0;
    float newsalary;
    printf("enter the employee id ");
    scanf("%d",&id);
    for(int i=0;i<count;i++){
        if(emp[i].emp_id==id){
            printf("enter the updated salary%s",emp[i].name);
            scanf("%f",&newsalary);
            emp[i].salary=newsalary;
            printf("salary is updated successfully");
```

```
            fount=1;
            break;
        }
    }
    if(!fount){
        printf("employe with %d id is not found",id);

    }
}
void display(struct Employee emp[],int count){
    if(count == 0){
        printf("there is no details");
    }
    else{
        printf("the employee details");
        for(int i=0;i<count;i++){
            printf("NAME:%s,ID:%d,SALARY:%.2f\n",emp[i].name,emp[i].emp_id,emp[i].salary);
        }
    }

}
void heighestsalary(struct Employee emp[],int count)
{
    if(count ==0){
        printf("there is no details");
    }
    else{
        int maxindex=0;
        for(int i=0;i<count;i++){
            if(emp[i].salary>emp[maxindex].salary){
                maxindex=i;
            }
        }
        printf("Employee with the highest salary:\n");
        printf("ID: %d, Name: %s, Salary: %.2f\n",emp[maxindex].emp_id, emp[maxindex].name,
emp[maxindex].salary);
    }
}
```

Problem 2: Library Management System

Objective: Manage a library system with a structure to store book details.
Description:
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
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 Library
{
   int book_id;
   char title[100];
   char author[50];
   int copies;


};
void add(struct Library lib[],int *count);
void issue(struct Library lib[],int count);
void returned(struct Library lib[],int count);
void search(struct Library lib [],int count);

int main(){
   struct Library lib[100];
   int count=0,op;



   while(1){
      printf("Library management function\n");
      printf("1 add books\n");
      printf("2 issue books\n");
      printf("3 return book\n");
      printf("4 searcha book\n");
      printf("choice an operation");
```

```
        scanf("%d",&op);
        switch (op)
        {
        case 1:
          add(lib,&count);
            break;
        case 2:
          issue(lib,count);
            break;
        case 3:
          returned(lib,count);
            break;
        case 4:
          search(lib,count);
            break;

        default:
            break;
        }

    }
}
void add(struct Library lib[],int *count){
    printf("enter the book name");
    scanf("%s",lib[*count].title);
    printf("enter the book id");
    scanf("%d",&lib[*count].book_id);
    printf("enter the author name");
    scanf("%s",lib[*count].author);
    printf("enter the copies available");
    scanf("%d",&lib[*count].copies);
    (*count)++;
    printf("the details is added succesfully");

}

void issue(struct Library lib[],int count){
    char title[100];
    printf("enter the book wanted");
    scanf("%s",title);
    for(int i=0;i<count;i++){
        if(strcmp(lib[i].title,title)==0){
            if(lib[i].copies>0){
                lib[i].copies--;
```

```
            printf("book issues sussesfully");
        }
        else{
            printf("no copies available");
        }
    }

  }
}

void returned(struct Library lib[],int count){
   char title[100];
   printf("enter the book returned");
   scanf("%s",title);
   for(int i=0;i<count;i++){
      if(strcmp(lib[i].title,title)==0){
            lib[i].copies++;
            printf("book returned sussesfully");
      }
      else{
         printf("there is no such books");
      }
   }
}
void search(struct Library lib [],int count){
   char search[100];
   int isfount =0;
   printf("enter the book name");
   scanf("%s",search);
   for(int i=0;i<count;i++){
      if(strcmp(lib[i].author,search)==0||strcmp(lib[i].title,search)==0){
          printf("ID: %d, Title: %s, Author: %s, Copies: %d\n",
              lib[i].book_id, lib[i].title, lib[i].author, lib[i].copies);
         isfount = 1;
      }
   }
   if(!isfount){
      printf("there is no such books");
   }
}
```

Problem 3: Cricket Player Statistics
Objective: Store and analyze cricket player performance data.
Description:
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
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.

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

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

void add(struct cricket player[], int *count);
void battleaverage(struct cricket player[], int count);
void heighestaverage(struct cricket player[], int count);

int main() {
    struct cricket player[100];
    int count = 0;
    int op;

    while (1) {
        printf("\nCricket Player Statistics\n");
        printf("1. Add Player Details\n");
        printf("2. Display Player Averages\n");
        printf("3. Display Player with Highest Average\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &op);
```

```
        switch (op) {
            case 1:
                add(player, &count);
                break;
            case 2:
                battleaverage(player, count);
                break;
            case 3:
                heighestaverage(player, count);
                break;
            case 4:
                printf("Exiting the program.\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
}

void add(struct cricket player[], int *count) {
    printf("\nEnter Player Details:\n");
    printf("Name: ");
    scanf(" %[^\n]", player[*count].name);
    printf("Number of Matches Played: ");
    scanf("%d", &player[*count].matches);
    printf("Total Runs Scored: ");
    scanf("%d", &player[*count].runs);

    if (player[*count].matches > 0) {
        player[*count].average = (float)player[*count].runs / player[*count].matches;
    } else {
        player[*count].average = 0.0;
    }

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

void battleaverage(struct cricket player[], int count) {
    if (count == 0) {
        printf("No player details available.\n");
        return;
    }
```

```
    printf("\nPlayer Statistics:\n");
    printf("Name\t\tMatches\t\tRuns\t\tAverage\n");
    for (int i = 0; i < count; i++) {
        printf("%-15s %-10d %-10d %.2f\n",
               player[i].name, player[i].matches, player[i].runs, player[i].average);
    }
}

void heighestaverage(struct cricket player[], int count) {
    if (count == 0) {
        printf("No player details available.\n");
        return;
    }

    int maxindex = 0;
    for (int i = 1; i < count; i++) {
        if (player[i].average > player[maxindex].average) {
            maxindex = i;
        }
    }

    printf("\nPlayer with the Highest Batting Average:\n");
    printf("Name: %s\n", player[maxindex].name);
    printf("Matches: %d\n", player[maxindex].matches);
    printf("Runs: %d\n", player[maxindex].runs);
    printf("Average: %.2f\n", player[maxindex].average);
}
```

Problem 4: Student Grading System
Objective: Manage student data and calculate grades based on marks.
Description:
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
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>

// Define the student structure
struct student {
    int rollno;
    char name[50];
    float mark[5];
    char grade;
};

// Function prototypes
void addstudent(struct student grade[], int *count);
void averagemark(struct student grade[], int count);
char assignGrade(float average);
void displaystudents(struct student grade[], int count);

int main() {
    struct student grade[100];
    int count = 0;
    int choice;

    while (1) {
        printf("\nStudent Grading System\n");
        printf("1. Add Student Details\n");
        printf("2. Calculate Grades\n");
        printf("3. Display Student Details\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addstudent(grade, &count);
                break;
            case 2:
                averagemark(grade, count);
                break;
            case 3:
                displaystudents(grade, count);
                break;
```

```
            case 4:
                printf("Exiting the program.\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void addstudent(struct student grade[], int *count) {
    printf("\nEnter Student Details:\n");
    printf("Roll Number: ");
    scanf("%d", &grade[*count].rollno);
    printf("Name: ");
    scanf(" %[^\n]", grade[*count].name);
    printf("Enter marks in 5 subjects:\n");
    for (int i = 0; i < 5; i++) {
        printf("Subject %d: ", i + 1);
        scanf("%f", &grade[*count].mark[i]);
    }
    (*count)++;
    printf("Student details added successfully.\n");
}

void averagemark(struct student grade[], int count) {
    if (count == 0) {
        printf("No students to calculate grades.\n");
        return;
    }

    for (int i = 0; i < count; i++) {
        float total = 0.0;
        for (int j = 0; j < 5; j++) {
            total += grade[i].mark[j];
        }
        float average = total / 5.0;
        grade[i].grade = assignGrade(average);
    }

    printf("Grades calculated successfully.\n");
}
char assignGrade(float average) {
```

```
    if (average >= 90)
        return 'A';
    else if (average >= 75)
        return 'B';
    else if (average >= 60)
        return 'C';
    else if (average >= 50)
        return 'D';
    else
        return 'F';
}

// Function to display student details
void displaystudents(struct student grade[], int count) {
    if (count == 0) {
        printf("No student details available.\n");
        return;
    }


    for (int i = 0; i < count; i++) {
        printf("%d %s ", grade[i].rollno, grade[i].name);
        for (int j = 0; j < 5; j++) {
            printf("%.2f ", grade[i].mark[j]);
        }
        printf("\t%c\n", grade[i].grade);
    }
}
```

Problem 5: Flight Reservation System
Objective: Simulate a simple flight reservation system using structures.
Description:
Define a structure Flight with fields:

char flight_number[10]: Flight number
char destination[50]: Destination city
int available_seats: Number of available seats
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.
has context menu

```
#include<stdio.h>
#include<string.h>
struct flight{
  char flight_number[10];
  char destination[50];
  int seats;

};
int main(){
   struct flight f1[20];
   int choice;
   int count=0;
   while(1){
      printf("1.Add flights\n");
      printf("2.book flights\n");
      printf("3.display flight details\n");
      printf("4.cancel tickets\n");
      printf("5.exit\n");
      printf("enter choice:");
      scanf("%d",&choice);
      switch(choice){
         case 1:
            printf("Flight number: ");
            getchar();
            scanf("%[^\n]", f1[count].flight_number);

            printf("Enter your destination: ");
            getchar();
            scanf("%[^\n]", f1[count].destination);

            printf("Enter number of seats: ");
            scanf("%d", &f1[count].seats);

            count++;
```

```
    printf("Flight added successfully!\n");
    printf("\n");
    break;

case 2:
     char flight[30];
    printf("Enter your flight number: ");
    getchar();
    scanf("%[^\n]", flight);

    //int booked = 0;

    for (int i = 0; i < count; i++) {
       if (strcmp(flight, f1[i].flight_number) == 0) {

            f1[i].seats--;
            printf("Flight booked successfully! Remaining seats: %d\n", f1[i].seats);
          } else {
            printf("No seats available on flight %s.\n", f1[i].flight_number);
          }

        }
         break;
         printf("\n");



 case 3:
     char dest[40];
     printf("enter destination:");
     getchar();
     scanf("%[^\n]",dest);

     for(int i=0;i<count;i++){
        if(strcmp(dest,f1[i].destination)==0){
          printf("Flight Number:%s\t seats:%d",f1[i].flight_number,f1[i].seats);
        }else{
            printf("enter valid destination\n");
        }

     }
     printf("\n");
     break;
```

```
        case 4:

            char flight1[30];
            printf("Enter your flight number: ");
            getchar();
            scanf("%[^\n]", flight1);



            for (int i = 0; i < count; i++) {
                if (strcmp(flight1, f1[i].flight_number) == 0) {

                    f1[i].seats++;
                    printf("Flight cancelled successfully! Remaining seats: %d\n", f1[i].seats);
                } else {
                    printf("No seats available on flight %s.\n", f1[i].flight_number);
                }

            }
             break;
             printf("\n");

        case 5:

            printf("EXIT\n");
    }


  }


   return 0;
}
```

Problem 1: Dynamic Array Resizing
Objective: Write a program to dynamically allocate an integer array and allow the user to resize it.
Description:
The program should ask the user to enter the initial size of the array.
Allocate memory using malloc.
Allow the user to enter elements into the array.
Provide an option to increase or decrease the size of the array. Use realloc to adjust the size.
Print the elements of the array after each resizing operation.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int *ptr =NULL;
    int n ,choice,newSize;
    printf("enter the element");
    scanf("%d",&n);

    ptr = (int*)malloc(n*sizeof(int));

    if(ptr==NULL){
        printf("the memory is not allocated");
    }
    else{
        printf("the memory is allocated");
    }
    for(int i=0;i<n;i++){
        scanf("%d",&ptr[i]);

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

     while (1) {
        printf("\nChoose an option:\n");
        printf("1. Increase array size\n");
        printf("2. Decrease array size\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
        if (choice == 3) {
            break;
        }
        printf("Enter the new size of the array: ");
        scanf("%d", &newSize);

        if (newSize <= 0) {
            printf("Invalid size. Array size must be greater than zero.\n");
            continue;
        }
        int *temp = (int *)realloc(ptr, newSize * sizeof(int));
        if (temp == NULL) {
            printf("Memory reallocation failed. Exiting.\n");
            free(ptr);
            return 1;
        }
        ptr = temp;

        if (newSize > n) {
            printf("Enter %d new elements:\n", newSize - n);
            for (int i = n; i < newSize; i++) {
                scanf("%d", &ptr[i]);
            }
        }
        n = newSize;
        printf("Updated array elements: ");
        for (int i = 0; i < n; i++) {
            printf("%d ", ptr[i]);
        }
        printf("\n");
    }

    free(ptr);
    printf("Memory successfully freed. Program terminated.\n");

    return 0;


}
```

Problem 2: String Concatenation Using Dynamic Memory
Objective: Create a program that concatenates two strings using dynamic memory allocation.
Description:
Accept two strings from the user.
Use malloc to allocate memory for the first string.
Use realloc to resize the memory to accommodate the concatenated string.
Concatenate the strings and print the result.
Free the allocated memory.

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

int main() {
    char *str1 = NULL;
    char *str2 = NULL;
    int ln, lm;

    printf("Enter the length of the first string: ");
    scanf("%d", &ln);
    str1 = (char *)malloc((ln + 1) * sizeof(char));
    if (str1 == NULL) {
        printf("Memory allocation failed for the first string.\n");
        return 1;
    }
    printf("Enter the first string: ");
    scanf(" %[^\n]", str1);

    printf("Enter the length of the second string: ");
    scanf("%d", &lm);
    str2 = (char *)malloc((lm + 1) * sizeof(char));
    if (str2 == NULL) {
        printf("Memory allocation failed for the second string.\n");
        free(str1);
        return 1;
    }
    printf("Enter the second string: ");
    scanf(" %[^\n]", str2);

    str1 = (char *)realloc(str1, (ln + lm + 1) * sizeof(char));
    if (str1 == NULL) {
```

```
        printf("Memory reallocation failed.\n");
        free(str2);
        return 1;
    }

    strcat(str1, str2);

    printf("Concatenated string: %s\n", str1);
    free(str1);
    free(str2);

    return 0;
}
```

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

typedef struct
{
    int row;
    int col;
    int val;
}s_matrix;

int main()
{
    int m, n, count=0;
    printf("Enter the number of rows and columns of the matrix: ");
    scanf("%d %d", &m, &n);

    int** matrix = (int**)malloc(m * sizeof(int *));
    for (int i = 0; i < m; i++)
    {
        matrix[i] = (int*)malloc(n * sizeof(int));
    }

    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < m; i++)
    {
```

```
        for (int j = 0; j < n; j++)
        {
            scanf("%d", &matrix[i][j]);
            if (matrix[i][j] != 0)
            {
                count++;
            }
        }
    }

    s_matrix *sparse_mat = (s_matrix *)malloc(count * sizeof(s_matrix));
    int k = 0;

    for(int i=0; i<m; i++)
    {
        for(int j=0; j<n; j++)
        {
            if(matrix[i][j] != 0)
            {
                sparse_mat[k].row = i;
                sparse_mat[k].col = j;
                sparse_mat[k].val = matrix[i][j];
                k++;
            }
        }
    }

    printf("\nSparse Matrix Representation:\n");
    printf("Row\tColumn\tValue\n");
    for (int i = 0; i < count; i++)
    {
        printf("%d\t%d\t%d\n", sparse_mat[i].row, sparse_mat[i].col, sparse_mat[i].val);
    }

    for (int i = 0; i < m; i++)
    {
        free(matrix[i]);
    }
    free(matrix);
    free(sparse_mat);
}
```

Objective: Write a program to dynamically allocate a 2D array.
Description:
Accept the number of rows and columns from the user.
Use malloc (or calloc) to allocate memory for the rows and columns dynamically.
Allow the user to input values into the 2D array.
Print the array in matrix format.
Free all allocated memory at the end.

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

int main() {
    int row, col;
    int **matrix;

    printf("Enter the number of rows: ");
    scanf("%d", &row);
    printf("Enter the number of columns: ");
    scanf("%d", &col);

    matrix = (int **)malloc(row * sizeof(int *));
    if (matrix == NULL) {
        printf("Memory allocation failed for row pointers.\n");
        return 1;
    }
    for (int i = 0; i < row; i++) {
        matrix[i] = (int *)malloc(col * sizeof(int));
        if (matrix[i] == NULL) {
            printf("Memory allocation failed for row %d.\n", i);
            for (int j = 0; j < i; j++) {
                free(matrix[j]);
            }
            free(matrix);
            return 1;
        }
    }
    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
```

```
            printf("matrix[%d][%d]: ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }
    printf("\nThe %d x %d matrix is:\n", row, col);
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
    for (int i = 0; i < row; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;
}
```

CLASS WORKS

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

int main(){
    int *ptr =NULL;
    int n;
    printf("enter the number integer");
    scanf("%d",&n);

    ptr =(int *)calloc(n,sizeof(int));

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

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

```
#include<stdlib.h>
#include<string.h>
int main(){
   char *str = NULL;
   // malloc
   str = (char*)calloc(15,sizeof(char));
   strcpy(str,"anus");
   printf("string =%s,address = %u\n",str,str);

   // realocating memory
   str =(char*)realloc(str,25);
   strcat(str,"ree");
   printf("string =%s,address=%u",str,str);

   free(str);
   return 0;
}
```

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

int main(){

   int **ipp;
   int i=4,j=5,k=6;
   int *ip1,*ip2;
   ip1=&i;
   ip2=&j;

   ipp=&ip1;
   printf("the value of ip1 %d\n",*ip1);
   printf("the value in ipp %d\n",**ipp);

   **ipp=8;
      printf("the value in ipp %d\n",**ipp);
        printf("the value in ipp %d\n",i);




}
```

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

```
int main(){
   int **ipp;
   int i=4,j=5,k=6;
   int *ip1,*ip2;

   ip1=&i;
   ip2=&j;
   ipp=&ip1;
   printf("the value is %d\n",**ipp);
   // afther modification of the value
   *ipp=ip2;
   printf("the value is %d\n",**ipp);
}
```

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main(){
   int * str =NULL;
   int num;

   printf("enter the number of elements");
   scanf("%d",&num);
   printf("\n");

   printf("the number entered is n=%d\n",num);
   // dynamically allocated memmory from the array

   str = (int*)malloc(num*sizeof(int));

   // check wheather the memory allocated succesfully or not

   if(str == NULL){
      printf("memory not allocated");
     exit(0);
   }
   else{
      printf("memory is allocated successfully");

   }
   // populating the array
   for(int i=0;i<num;i++){
         str[i]=i+1;
      }
```

```
    // displaying the array
    for(int i=0;i<num;i++){
            printf("%d",str[i]);
        }
    free(str);
    return 0;
}
```

```
#include <stdio.h>


struct currentDate{

    int day;

    int month;

    int year;

};

struct currentTime{

    int sec;

    int min;

    int hours;

};

struct CDateTime{

    struct currentDate d1;

    struct currentTime t1;

};

int main(){

    struct CDateTime dt = {{21, 11, 2024}, {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.hours);

    return 0;

}
-------------------------------------------
#include<stdio.h>
struct data{
    int day;
    int month;
    int year;
}today;
int main(){
    // struct data today;
    today.day=21;
    today.month=11;
    today.year=2024;
    printf("todays date is %d-%d-%d\n",today.day,today.month,today.year);
    printf("the today gives%d",sizeof(today));
}
-----------------------------------------------------------------
#include<stdio.h>
struct data{
    int day;
    int month;
    int year;
}today,tom;
int main(){
    // struct data today;
    today.day=21;
    today.month=11;
    today.year=2024;
    struct data today={21,4,2002};

    printf("todays date is %d-%d-%d\n",today.day,today.month,today.year);
    printf("the today gives%d\n",sizeof(today));

    tom.day =8;
    tom.month=3;
    tom.year=2023;
    printf("todays date is %d-%d-%d\n",tom.day,tom.month,tom.year);

}
```

```
---------------------------------------------------------------------------
#include<stdio.h>
struct data{
    int day;
    int month;
    int year;
};
int main(){
    // struct data today;
    // today.day=21;
    // today.month=11;
    // today.year=2024;
    struct data today={21,4,2002};

    printf("todays date is %d-%d-%d\n",today.day,today.month,today.year);
    printf("the today gives%d\n",sizeof(today));

    // tom.day =8;
    // tom.month=3;
    // tom.year=2023;
    // printf("todays date is %d-%d-%d\n",tom.day,tom.month,tom.year);

}
-----------------------------------------------------------------------
#include <stdio.h>

struct student{

    char name[50];

    int rollNumber;

    float marks;

};

int main(){

    //struct date today;

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

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

```
    return 0;

-----------------------------------------------------------------------------------------
#include<stdio.h>
struct Coordinate{
    int x;
    int y;
};

void printCoordinate(struct Coordinate);

int main(){
    // struct Coordinate pointA={5,6};
    // printCoordinate(pointA);
    printCoordinate((struct Coordinate){5,6});

}

void printCoordinate(struct Coordinate temp){
    printf("x=%d y=%d\n",temp.x,temp.y);

}
-------------------------------------------------------------------
#include<stdio.h>
struct Coordinate{
    int x;
    int y;
};


int main(){
   struct Coordinate pnt[5];

   for(int i=0;i<5;i++){
    printf("initialize the struct present in the %d\n",i);
    scanf("%d %d",&pnt[i].x,&pnt[i].y);
   }

   for(int i=0;i<5;i++){
    printf("display the coordinates at index %d value is  %d  %d\n",i,pnt[i].x,pnt[i].y);
   }

}
```

```
---------------------------------------------------------------------
#include<stdio.h>
struct Coordinate{
    int x;
    int y;
    int z;
};


int main(){
   struct Coordinate pnt[5]={12,10,1975,12,30,1980,11,15,2005};

  for(int i=0;i<3;i++){
    printf("display the coordinates at index %d value is  %d  %d  %d\n",i,pnt[i].x,pnt[i].y,pnt[i].z);
   }


}
```