Problem : Dynamic Student Record Management

Objective: Manage student records using pointers to structures and dynamically allocate memory for student names.

Description:

Define a structure Student with fields:

int roll_no: Roll number

char *name: Pointer to dynamically allocated memory for the student's name

float marks: Marks obtained

Write a program to:

Dynamically allocate memory for n students.

Accept details of each student, dynamically allocating memory for their names.

Display all student details.

Free all allocated memory before exiting.

```
#include<stdio.h>
#include<stdlib.h>
struct student{
  int roll_no;
  char *name;
  float marks;

};
int main(){
    int n;
    printf("enter no of students:");
    scanf("%d",&n);

    struct student *ptr=(struct student*)malloc(n*sizeof(struct student));
```

```
    //printf("enter student details\n");

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

        ptr[i].name=(char*)malloc(100*sizeof(char));

        printf("enter student name:");
        scanf("%s",ptr[i].name);



        printf("enter roll no:");
        scanf("%d",&ptr[i].roll_no);

        printf("enter marks:");
        scanf("%f",&ptr[i].marks);
         }

    for(int i=0;i<n;i++){
        printf("%d\t%s\t%.2f\n",ptr[i].roll_no,ptr[i].name,ptr[i].marks);
    }

    for (int i = 0; i < n; i++) {
        free(ptr[i].name);
    }
    free(ptr);



return 0;
}
```

Problem : Library System with Dynamic Allocation

Objective: Manage a library system where book details are dynamically stored using pointers inside a structure.

Description:

Define a structure Book with fields:

char *title: Pointer to dynamically allocated memory for the book's title

char *author: Pointer to dynamically allocated memory for the author's name

int *copies: Pointer to the number of available copies (stored dynamically)

Write a program to:

Dynamically allocate memory for n books.

Accept and display book details.

Update the number of copies of a specific book.

Free all allocated memory before exiting.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct libary{
   char *title;
   char *author;
   int *copies;

};

int main(){
   int n;
   printf("enter no of books:");
   scanf("%d",&n);

   struct libary *ptr=(struct libary*)malloc(n*sizeof(struct libary));
```

```
for(int i=0;i<n;i++){
  ptr[i].title=(char*)malloc(100*sizeof(char));
  ptr[i].author=(char*)malloc(100*sizeof(char));
  ptr[i].copies=(int*)malloc(sizeof(int));


  printf("enter book name:");
  scanf("%s",ptr[i].title);

   printf("enter naame of the author :");
  scanf("%s",ptr[i].author);


   printf("enter no of copies:");
  scanf("%d",ptr[i].copies);
  }

   for(int i=0;i<n;i++){
   printf("Book Name:%s\tAuthor Name:%s\tCopies:%d\n",ptr[i].title,ptr[i].author,*ptr[i].copies);
   }

   for(int i=0;i<n;i++){
      char book[30];
      printf("enter name of the book to update copies:");
      getchar();
      scanf("%[^\n]",book);
      if(strcmp(book,ptr[i].title)==0){
         (*ptr[i].copies)++;
         printf("Updated number of copies for '%s': %d\n", ptr[i].title, *ptr[i].copies);
      }
```

```
    }

    for(int i=0;i<n;i++){
        free(ptr[i].title);
        free(ptr[i].author);
        free(ptr[i].copies);
    }
    free(ptr);
}
```

Problem 1: Complex Number Operations

Objective: Perform addition and multiplication of two complex numbers using structures passed to functions.

Description:

Define a structure Complex with fields:

float real: Real part of the complex number

float imag: Imaginary part of the complex number

Write functions to:

Add two complex numbers and return the result.

Multiply two complex numbers and return the result.

Pass the structures as arguments to these functions and display the results.

```
#include<stdio.h>
struct complex{
   float real;
   float image;
};

void add(struct complex,struct complex);
void multiply(struct complex a,struct complex b);
int main(){
```

```
    struct complex sum;
    struct complex number1={3,4};
    struct complex number2={2,5};
    add(number1,number2);
    multiply(number1,number2);

    return 0;
}

 void add(struct complex a,struct complex b)
{
    struct complex result;
    result.real=a.real+b.real;
    result.image=a.image+b.image;
    printf("result of addition= %.2f+%.2fi\n",result.real,result.image);
}

void multiply(struct complex a,struct complex b){
    struct complex result;
    result.real = (a.real * b.real) - (a.image * b.image);
    result.image = (a.real * b.image) + (a.image * b.real);

    printf("result of multiplication = %.2f+%.2fi",result.real,result.image);

}
```

Problem : Rectangle Area and Perimeter Calculator

Objective: Calculate the area and perimeter of a rectangle by passing a structure to functions.

Description:

Define a structure Rectangle with fields:

float length: Length of the rectangle

float width: Width of the rectangle

Write functions to:

Calculate and return the area of the rectangle.

Calculate and return the perimeter of the rectangle.

Pass the structure to these functions by value and display the results in main.

```
#include<stdio.h>
#include<math.h>
struct rectangle{
   float length;
   float width;

};
void area(struct rectangle);
void perimeter(struct rectangle);

int main(){
   struct rectangle Rectangle={5,3};
   area(Rectangle);
   perimeter(Rectangle);

}

void area(struct rectangle a){
   printf("Area=%.2f\n",a.length*a.width);
}

void perimeter(struct rectangle a){

   printf("Perimeter= %.2f\n",2*(a.length+a.width));}
```

Problem : Student Grade Calculation

Objective: Calculate and assign grades to students based on their marks by passing a structure to a function.

Description:

Define a structure Student with fields:

char name[50]: Name of the student

int roll_no: Roll number

float marks[5]: Marks in 5 subjects

char grade: Grade assigned to the student

Write a function to:

Calculate the average marks and assign a grade (A, B, etc.) based on predefined criteria.

Pass the structure by reference to the function and modify the grade field.

```
#include<stdio.h>

struct student{
   char name[50];
   int roll_no;
   float marks[5];
};
void student_grade(struct student);
int main(){

   struct student newStudent;
   printf("Enter name");
   scanf("%s",&newStudent.name);
   printf("enter roll number");
   scanf("%d",&newStudent.roll_no);
   printf("enter marks obtained in 5 subjects");
   for(int i=0;i<5;i++){
```

```
        scanf("%f",&newStudent.marks[i]);
    }

    printf("STUDENT DETAILS:\n");
    printf("name : %s\n",newStudent.name);
    printf("roll no : %d\n",newStudent.roll_no);
    printf("marks\n");
    for(int i=0;i<5;i++){
        printf("%.2f\t",newStudent.marks[i]);
    }printf("\n");

    printf("Your Grade is:\n");
    student_grade(newStudent);

}

void student_grade(struct student new){
    float sum=0,average;
    for(int i=0;i<5;i++){
        sum=sum+new.marks[i];

    }average=sum/5;
    printf("your average mark in 5 subject is %.2f\t: ",average);

    if(average>=90){
        printf("GRADE A");
    }else if(average>=80&&average<90){
        printf("GRADE B");
    }else if(average>=70&&average<80){
        printf("GRADE C");
    }else if(average>=60&&average<70){
```

```
        printf("GRADE D");


    }else if(average<60&&average>=0){
        printf("GRADE F");
    }else{
        printf("check the etered grade");
    }

}
```

Problem 4: Point Operations in 2D Space

Objective: Calculate the distance between two points and check if a point lies within a circle using structures.

Description:

Define a structure Point with fields:

float x: X-coordinate of the point

float y: Y-coordinate of the point

Write functions to:

Calculate the distance between two points.

Check if a given point lies inside a circle of a specified radius (center at origin).

Pass the Point structure to these functions and display the results.

```
#include<stdio.h>
#include<math.h>
struct coordinates{
    int x;
    int y;
};
void find_distance(struct coordinates,struct coordinates);
void inside_circle_or_not(struct coordinates ,struct coordinates );
int main(){
```

```
    struct coordinates coord1={3,4};
    struct coordinates coord2={6,8};
    struct coordinates center={0,0};
    find_distance(coord1,coord2);
    inside_circle_or_not(coord1,center);
    inside_circle_or_not(coord2,center);
    // int num=sqrt(9);
    // printf("%d",num);
}

void find_distance(struct coordinates a,struct coordinates b){

    int distance;
    distance=sqrt(((b.x-a.x)*(b.x-a.x)) + ((b.y-a.y)*(b.y-a.y)));
    printf("Distance= %d\n",distance);

}

void inside_circle_or_not(struct coordinates a,struct coordinates b){
    int radius=5;
    int distance;
    distance=sqrt(((b.x-a.x)*(b.x-a.x)) + ((b.y-a.y)*(b.y-a.y)));
    if(distance>radius){
        printf("(%d,%d) are outside the circle\n",a.x,a.y);
    }else{
        printf("(%d,%d) are inside the circle\n",a.x,a.y);
    }

}
```

Problem 5: Employee Tax Calculation

Objective: Calculate income tax for an employee based on their salary by passing a structure to a function.

Description:

Define a structure Employee with fields:

char name[50]: Employee name

int emp_id: Employee ID

float salary: Employee salary

float tax: Tax to be calculated (initialized to 0)

Write a function to:

Calculate tax based on salary slabs (e.g., 10% for salaries below $50,000, 20% otherwise).

Modify the tax field of the structure.

Pass the structure by reference to the function and display the updated tax in main.

```
#include<stdio.h>
struct employee{
   char name[50];
   int id;
   float salary;
   float tax;
};

void calculate_tax(struct employee*);

int main(){
   struct employee *emp;
   printf("enter name");
   scanf("%s",&emp->name);
   printf("enter emp id");
   scanf("%d",&emp->id);
   printf("enter salary");
```

```
    scanf("%f",&emp->salary);

    calculate_tax(emp);
}

void calculate_tax(struct employee *ptr){
    if(ptr->salary<50000&&ptr->salary>0){
        ptr->tax=ptr->salary*0.10;
        printf("Tax=%.2f",ptr->tax);
    }else{
        ptr->tax=ptr->salary*0.20;
         printf("Tax=%.2f",ptr->tax);
    }

}
```

Problem Statement: Vehicle Service Center Management

Objective: Build a system to manage vehicle servicing records using nested structures.

Description:

Define a structure Vehicle with fields:

char license_plate[15]: Vehicle's license plate number

char owner_name[50]: Owner's name

char vehicle_type[20]: Type of vehicle (e.g., car, bike)

Define a nested structure Service inside Vehicle with fields:

char service_type[30]: Type of service performed

float cost: Cost of the service

char service_date[12]: Date of service

Implement the following features:

Add a vehicle to the service center record.

Update the service history for a vehicle.

Display the service details of a specific vehicle.

Generate and display a summary report of all vehicles serviced, including total revenue.

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


struct Service{
   char service_type[30];
   float cost;
   char service_date[12];
};

struct Vehicle{
   char license_plate[15];
   char owner_name[50];
   char vehicle_type[20];
   struct Service Service_history[10];

};
struct Vehicle Vehicles[10];
struct Service Services[10];

int vehicle_count=0,service_count=0;
void addVehicle();
void update_service_history();
void service_details();
void all_history();

int main(){

   int op;
```

```
    while(1){
        printf("enter option\n1.\n2\n3.\n4.\n");
        scanf("%d",&op);

        switch (op)
        {
        case 1:
        addVehicle();

            break;
        case 2:
        update_service_history();

            break;
         case 3:
       service_details();

            break;
            case 4:
       all_history();

            break;

        default:
            break;
        }

    }

    return 0;
}
```

```
void addVehicle(){
    struct Vehicle a;
    printf("Enter license plate number:");
    scanf("%s",&a.license_plate);

    printf("Enter owner name:");
    scanf("%s",&a.owner_name);

    printf("Enter vehicle type:");
    scanf("%s",&a.vehicle_type);

    Vehicles[vehicle_count]=a;
    vehicle_count++;
    printf("Vechicle added successfully....\n");

}

void update_service_history(){
    char license[50];

    printf("enter lisense number");
    scanf("%s",&license);
    for(int i=0;i<vehicle_count;i++){
        if(strcmp(license,Vehicles[i].license_plate)==0){
            printf("Enter service type");
            scanf("%s",&Vehicles[i].Service_history.);
            printf("Enter vehicle cost");
            scanf("%f",&Vehicles[i].Service_history->cost);
            printf("Enter date of service:");
            scanf("%s",&Vehicles[i].Service_history->service_date);
```

```
            printf("updated succesfuly\n");
            printf("\n");


        }

    }

}

void service_details(){
    char license[50];
    printf("enter lisense number");
    scanf("%s",&license);
    for(int i=0;i<vehicle_count;i++){
        if(strcmp(license,Vehicles[i].license_plate)==0){
            printf("LICENSE PLATE : %s\n",Vehicles[i].license_plate);

            printf("OWNER NAME : %s\n",Vehicles[i].owner_name);

            printf("VEHICLE TYPE : %s\n",Vehicles[i].vehicle_type);

            printf("SERVICE HISTORY\n");
            printf("service type: %s\n",Vehicles[i].Service_history->service_type);
            printf("COST: %2.f\n",Vehicles[i].Service_history->cost);
            printf("DATE OF SERVICE :%s\n",Vehicles[i].Service_history->service_date);


            printf("\n");
```

```
        }

    }

}

void all_history(){
    for(int i=0;i<vehicle_count;i++){
        printf("LICENSE PLATE : %s\n",Vehicles[i].license_plate);

            printf("OWNER NAME : %s\n",Vehicles[i].owner_name);

            printf("VEHICLE TYPE : %s\n",Vehicles[i].vehicle_type);

            printf("SERVICE HISTORY\n");
            printf("service type: %s\n",Vehicles[i].Service_history->service_type);
            printf("COST: %2.f\n",Vehicles[i].Service_history->cost);
            printf("DATE OF SERVICE :%s\n",Vehicles[i].Service_history->service_date);


            printf("\n");


    }

}
```