Problem 1: Dynamic Student Record Management

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

Description:

1. 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
2. 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.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct student
{
    int roll_no;
    char *name;
    float marks;
};
int main()
{
    int choice, num;
    struct student *st = NULL;
    int count = 0,count1=0;
    printf("1. Add a student\n2. Display all students\n3. Find student by roll number\n");
    printf("4. Calculate average marks\n5. Exit\n");

    do {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                if (count == count1)
                {
```

```
                    if(count1==0)
                    {
                        count1=5;
                    }
                    else
                    {
                        count1+=1;
                    }
                    st = realloc(st, count1 *sizeof(struct student));
                }
                printf("Enter name: \n");
                getchar();
                char temp[100];
                gets(temp);
                st[count].name = malloc(strlen(temp) + 1);
                strcpy(st[count].name, temp);
                printf("Enter roll number: \n");
                scanf("%d",&st[count].roll_no);
                printf("Enter marks: \n");
                scanf("%f",&st[count].marks);
                count++;
                break;

            case 2:
                if (count == 0)
                {
                    printf("No students\n");
                }
                else
                {
                    for (int i = 0; i < count; i++)
                    {
                        printf("Student %d:\n", i + 1);
                        printf("Name: %s\n", st[i].name);
                        printf("Roll Number: %d\n", st[i].roll_no);
                        printf("Marks: %.2f\n", st[i].marks);
                        printf("\n");
                    }
                }
                break;

            case 3:
                printf("Enter student roll number: \n");
                scanf("%d", &num);
                int found = 0;
```

```
                for (int i = 0; i < count; i++)
                {
                    if (st[i].roll_no == num)
                    {
                        printf("Student found:\n");
                        printf("Name: %s\n", st[i].name);
                        printf("Roll Number: %d\n", st[i].roll_no);
                        printf("Marks: %.2f\n", st[i].marks);
                        found = 1;
                        break;
                    }
                }
                if (!found)
                {
                    printf("Student with roll number %d not found.\n",
   num);
                }
                break;

            case 4:
                    float avg = 0;
                    for (int i = 0; i < count; i++)
                    {
                        avg += st[i].marks;
                    }
                    avg /= count;
                    printf("Average marks: %.2f\n", avg);

                      break;

                  case 5:
                      printf("Exiting system.\n");


                      for (int i = 0; i < count; i++)
                      {
                          free(st[i].name);
                      }
                      free(st);
                      exit(0);

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

```
        } while (1);

        return 0;
    }

```

Problem 2: Library System with Dynamic Allocation

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

Description:

1. 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)
2. 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.

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

struct book
{
    int book_id;
    char *title;
    char *author;
    int *copies;
};
int main() {
    int choice, num;
    struct book *books = NULL;
    char name[50];
    int count = 0, count1 = 0;
    printf("Library System Options:\n");
```

```
20.     printf("1. Add books to the library.\n");
21.     printf("2. Issue a book.\n");
22.     printf("3. Return a book.\n");
23.     printf("4. Search for a book by title or author name.\n");
24.     printf("5. Exit.\n");
25.
26.     do {
27.         printf("\nEnter your choice: ");
28.         scanf("%d", &choice);
29.         switch (choice)
30.         {
31.             case 1:
32.                 if (count == count1)
33.                 {
34.                     if (count1 == 0)
35.                         count1 = 5;
36.                     else
37.                         count1 *= 1;
38.                     books = realloc(books, count1 * sizeof(struct book));
39.                 }
40.                 printf("Enter the book ID: ");
41.                 scanf("%d", &books[count].book_id);
42.                 getchar();
43.                 printf("Enter the title: \n");
44.                 char temp[100];
45.                 gets(temp);
46.                 books[count].title = malloc(strlen(temp) + 1);
47.                 strcpy(books[count].title, temp);
48.                 printf("Enter the author name: \n");
49.                 char temp1[100];
50.                 gets(temp1);
51.                 books[count].author = malloc(strlen(temp1) + 1);
52.                 strcpy(books[count].author, temp1);
53.                 printf("Enter the number of copies: \n");
54.                 books[count].copies = malloc(sizeof(int));
55.                 scanf("%d", books[count].copies);
56.                 count++;
57.                 break;
58.
59.             case 2:
60.                 printf("Enter the book id to issue: \n");
61.                 scanf("%d", &num);
62.                 for (int i = 0; i < count; i++)
63.                 {
64.                     if (books[i].book_id == num)
```

```
                    {
                        if (*books[i].copies > 0)
                        {
                            (*books[i].copies)--;
                            printf("Book '%s' issued successfully.\n", books[i].title);
                        } else
                        {
                            printf("No copies available for '%s'.\n", books[i].title);
                        }
                        break;
                    }
                }
                break;

            case 3:
                printf("Enter the book id to return: \n");
                scanf("%d", &num);
                for (int i = 0; i < count; i++)
                {
                    if (books[i].book_id == num)
                    {
                        (*books[i].copies)++;
                        printf("Book '%s' returned successfully.\n", books[i].title);
                        break;
                    }
                }
                break;

            case 4:
                printf("Enter the book title or author name to search: ");
                getchar();
                fgets(name, sizeof(name), stdin);
                int found = 0;
                for (int i = 0; i < count; i++)
                {
                        if (strcmp(name, books[i].title) == 0 || strcmp(name, books[i].author) == 0)
                        {
                            printf("Book ID: %d\n", books[i].book_id);
                            printf("Title: %s\n", books[i].title);
                            printf("Author: %s\n", books[i].author);
```

```
                                printf("Copies Available: %d\n",
   *books[i].copies);
                                found = 1;
                            }
                        }
                        if (!found)
                        {
                            printf("No book found \n");
                        }
                        break;

                    case 5:
                        printf("Exiting the system.\n");
                        for (int i = 0; i < count; i++) {
                            free(books[i].title);
                            free(books[i].author);
                            free(books[i].copies);
                        }
                        free(books);
                        exit(0);

                    default:
                        printf("Invalid choice. Please try again.\n");
                        break;
                }
            } while (1);

            return 0;
        }
```

Problem 1: Complex Number Operations

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

Description:

1. Define a structure Complex with fields:
    - float real: Real part of the complex number
    - float imag: Imaginary part of the complex number
2. Write functions to:

- Add two complex numbers and return the result.
- Multiply two complex numbers and return the result.

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

```c
#include<stdio.h>
struct complex
{
    float real;
    float imag;
};
void add(struct complex, struct complex);
void mul(struct complex, struct complex);
int main()
{
    struct complex num1,num2;
    printf("enter the real and imaginary part of the first complex number \n");
    scanf("%f %f",&num1.real,&num1.imag);
    printf("enter the real and imaginary part of the second complex number \n");
    scanf("%f %f",&num2.real,&num2.imag);
    add(num1,num2);
    mul(num1,num2);
    return 0;
}
void add(struct complex n1, struct complex n2)
{
    printf("the added complex number is %0.2f+%0.2fi\n",n1.real+n2.real,n1.imag+n2.imag);
}
void mul(struct complex n1, struct complex n2)
{
    printf("the multiplied complex number is %0.2f+%0.2fi\n",n1.real*n2.real-n1.imag*n2.imag,n1.real*n2.imag+n1.imag*n2.real);
}
```

Problem 2: Rectangle Area and Perimeter Calculator

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

Description:

1. Define a structure Rectangle with fields:
    - float length: Length of the rectangle
    - float width: Width of the rectangle
2. Write functions to:
    - Calculate and return the area of the rectangle.
    - Calculate and return the perimeter of the rectangle.
3. Pass the structure to these functions by value and display the results in main.

```c
#include<stdio.h>
struct rectangle
{
    float length;
    float breadth;
};
float area(struct rectangle);
float per(struct rectangle);
int main()
{
    float Area,Per;
    struct rectangle rec1;
    printf("enter the length and breadth of the first rectangle\n");
    scanf("%f %f",&rec1.length,&rec1.breadth);
    Area=area(rec1);
    Per=per(rec1);
    printf("area of the rectangle is %0.2f \n",Area);
    printf("perimeter of the rectangle is %0.2f",Per);
    return 0;
}
float area(struct rectangle rec)
{
    float ar=rec.length*rec.breadth;
    return ar;
}
float per(struct rectangle rem)
{
    float  perimeter=2*(rem.length+rem.breadth);
    return perimeter;
}
```

Problem 3: Student Grade Calculation

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

Description:

1. 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
2. Write a function to:
    - Calculate the average marks and assign a grade (A, B, etc.) based on predefined criteria.
3. Pass the structure by reference to the function and modify the grade field.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct student
{
    char name[50];
    int roll_no;
    float marks[5];
    char grade;
};
void grade(struct student *);
int main()
{
    struct student st;
    printf("Enter the name of the student \n");
    gets(st.name);
    printf("enter the roll number\n");
    scanf("%d",&st.roll_no);
    printf("enter the marks of 5 subjects \n");
    for(int i=0;i<5;i++)
    {
        scanf("%f",&st.marks[i]);
    }
    grade(&st);
    return 0;
}
```

```c
void grade(struct student *s1)
{
    float avg=0;
    for(int i=0;i<5;i++)
    {
        avg+=s1->marks[i];
    }
    avg=avg/5;
    if(avg>=90)
        s1->grade='A';
    else  if(avg>=80 && avg<=90)
        s1->grade='B';
    else  if(avg>=70 && avg<=80)
        s1->grade='C';
    else
        s1->grade='D';
    printf("the grade of the student is %c ",s1->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:

1. Define a structure Point with fields:
    - float x: X-coordinate of the point
    - float y: Y-coordinate of the point
2. 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).
3. Pass the Point structure to these functions and display the results.

```c
#include <stdio.h>

#include <math.h>
struct point
```

```c
{
    float x;
    float y;
};
float distance(struct point p1, struct point p2);
int circle(struct point p, float radius) ;
int main()
{
    struct point pt1, pt2, check;
    float radius;
    printf("Enter the x and y coordinates of the first point: ");
    scanf("%f %f", &pt1.x, &pt1.y);
    printf("Enter the x and y coordinates of the second point: ");
    scanf("%f %f", &pt2.x, &pt2.y);
    float dist=distance(pt1, pt2);
    printf("The distance between the two points is: %.2f\n", dist);
    printf("Enter the x and y coordinates of the point to check: ");
    scanf("%f %f", &check.x, &check.y);
    printf("Enter the radius of the circle ");
    scanf("%f", &radius);
    if (circle(check, radius))
    {
        printf("The point  lies within the circle \n");
    } else {
        printf("The point does not lie within the circle \n");
    }
    return 0;
}
float distance(struct point p1, struct point p2)
{
    return sqrt((p2.x-p1.x)*(p2.x-p1.x)+(p2.y-p1.y)*(p2.y-p1.y));
}
int circle(struct point p, float radius)
{
    float distan = sqrt(p.x*p.x+p.y*p.y);
    if(distan <= radius)
        return 1;
}
```

Problem 5: Employee Tax Calculation

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

Description:

1. 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)
2. 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.
3. Pass the structure by reference to the function and display the updated tax in main.

has context menu

```
#include <stdio.h>
#include <math.h>
struct employee
{
    char name[50];
    int emp_id;
    float salary;
    float tax;
};
void taxes(struct employee *);
int main()
{
   struct employee emp;
   emp.tax=0;
    printf("enter the name of the employee \n");
    gets(emp.name);
    printf("enter the id \n");
    scanf("%d",&emp.emp_id);
    printf("enter the salary \n");
    scanf("%f",&emp.salary);
    taxes(&emp);
    printf("tax of employee %s with employee id %d is %0.2f
",emp.name,emp.emp_id,emp.tax);
    return 0;
}
void taxes(struct employee *em)
{
    if((*em).salary<=5000)
```

```
        (*em).tax=10;
    else
        (*em).tax=20;
}
```

Problem Statement: Vehicle Service Center Management

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

Description:

1. 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)
2. 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
3. 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 <math.h>
#include<string.h>
struct vehicle
{
    char license_plate[15];
   char owner_name[50];
   char vehicle_type[20];
};
struct service
{
    char service_type[30];
    float cost;
    char service_date[12];
};
struct combined
{
    struct vehicle v1;
```

```
22.     struct service v2;
23. };
24. int main()
25. {
26.     int choice;
27.     static int count=0;
28.      char name[10];
29.      int found;
30.     struct combined c[5];
31.     printf("Add a vehicle to the service center record.\nUpdate the
    service history for a vehicle.\nDisplay the service details of a specific
    vehicle.\nGenerate and display a summary report of all vehicles serviced,
    including total revenue.\n");
32.     do
33.     {
34.     printf("enter a choice \n");
35.     scanf("%d",&choice);
36.     switch(choice)
37.     {
38.         case 1:
39.             getchar();
40.             printf("enter license plate number \n");
41.             gets(c[count].v1.license_plate);
42.             printf("enter name of the owner \n");
43.             gets(c[count].v1.owner_name);
44.             printf("enter type of vehicle \n");
45.             gets(c[count].v1.vehicle_type);
46.             printf("enter type of service \n");
47.             gets(c[count].v2.service_type);
48.             printf("enter cost of vehicle \n");
49.             scanf("%f",&c[count].v2.cost);
50.             printf("enter service date \n");
51.             gets(c[count].v2.service_date);
52.             count++;
53.             break;
54.
55.         case 2:
56.             found=0;
57.             printf("enter the license plate number of the vehicle to be
    updated \n");
58.             gets(name);
59.             for(int i=0;i<count;i++)
60.             {
61.                 if(strcmp(name,c[i].v1.license_plate)==0)
62.                 {
```

```
                    printf("enter type of service \n");
                    gets(c[count].v2.service_type);
                    printf("enter type of vehicle \n");
                    scanf("%f",&c[count].v2.cost);
                    getchar();
                    printf("enter service date \n");
                    gets(c[count].v2.service_date);
                    found++;
                    break;
                }
            }
            if(!found)
                    printf("vehicle not found \n");
            break;

        case 3:
            found=0;
            printf("enter the license plate number of the vehicle to be updated \n");
            char name[10];
            gets(name);
            int found=0;
            for(int i=0;i<count;i++)
            {
                if(strcmp(name,c[i].v1.license_plate)==0)
                {
                    found=1;
                    printf("service history \n");
                    printf("service type= %s\ncost=%f\nservice date=%s\n",c[i].v2.service_type,c[i].v2.cost,c[i].v2.service_date);
                    break;
                }
            }
            if(!found)
            {
                printf("vehicle not found \n");
            }
            break;

              case 4:
                  printf("summary report \n");
                  for(int i=0;i<count;i++)
                  {
                      printf("license plate= %s\nowner name= %s\nvehicle type= %s\nservice type= %s\ncost=%f\nservice date=%s
```

```
   \n",c[i].v1.license_plate,c[i].v1.owner_name,c[i].v1.vehicle_type,c[i].v2.
   service_type,c[i].v2.cost,c[i].v2.service_date);
                }
                break;

            default:
                    printf("wrong choice \n");
                    break;
        }
        return 0;
        }
    }
```
