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.

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



struct Student
{
  int roll_no;
  char *name;
  float marks;
};
```

```
void acceptDetails(struct Student *students, int n);
void displayDetails(struct Student *students, int n);
void freeMemory(struct Student *students, int n);

int main()
{
   int n;
   printf("Enter the number of students: ");
   scanf("%d", &n);



   struct Student *students = (struct Student *)malloc(n * sizeof(struct Student));



   acceptDetails(students, n);
   displayDetails(students, n);
   freeMemory(students, n);

   return 0;
}



void acceptDetails(struct Student *students, int n)
{
   char temp[100];
   for (int i = 0; i < n; i++) {
      printf("Enter details for student %d:\n", i + 1);
```

```
        printf("Roll Number: ");
        scanf("%d", &students[i].roll_no);


        printf("Name: ");

        scanf("%s",temp);
        students[i].name = (char *)malloc(strlen(temp) + 1);

        strcpy(students[i].name, temp);

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


void displayDetails(struct Student *students, int n)
{
    printf("\n");
    for (int i = 0; i < n; i++)
    {
        printf("Student %d:\n", i + 1);
        printf("  Roll Number: %d\n", students[i].roll_no);
        printf("  Name: %s\n", students[i].name);
        printf("  Marks: %.2f\n", students[i].marks);
    }
    printf("\n");
```

```
}



void freeMemory(struct Student *students, int n)
{
  for (int i = 0; i < n; i++)
  {
    free(students[i].name);
  }
  free(students);
}
```

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.

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

struct Book
{
   char *title;
   char *author;
   int *copies;
};

void acceptDetails(struct Book *books, int n);
void displayDetails(struct Book *books, int n);
void updateCopies(struct Book *books, int n);
void freeMemory(struct Book *books, int n);

int main()
{
   int n,choice;
   int a=0;
   printf("Enter the number of books: ");
   scanf("%d", &n);

   struct Book *books = (struct Book *)malloc(n * sizeof(struct Book));
```

```
    acceptDetails(books, n);

    while(a!=1)
    {
        printf("1.display book details\n2.update copies\n3.free memory");
        printf("\n");
        printf("Enter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                displayDetails(books, n);
                break;

            case 2:
                updateCopies(books, n);
                break;

            case 3:
                freeMemory(books, n);
                a=1;
                break;
        }
    }
    return 0;
}

void acceptDetails(struct Book *books, int n)
```

```
{
    char temp[100];
    for (int i = 0; i < n; i++) {
        printf("Enter details for book %d:\n", i + 1);

        printf("Title: ");
        scanf("%s",temp);
        books[i].title = (char *)malloc(strlen(temp));
        strcpy(books[i].title, temp);

        printf("Author: ");
        scanf("%s",temp);
        books[i].author = (char *)malloc(strlen(temp));
        strcpy(books[i].author, temp);

        books[i].copies = (int *)malloc(sizeof(int));
        printf("Number of Copies: ");
        scanf("%d", books[i].copies);
    }
}


void displayDetails(struct Book *books, int n)
{
    printf("\nLibrary Book Details:\n");
    for (int i = 0; i < n; i++)
    {
        printf("Book %d:\n", i + 1);
```

```
        printf("  Title: %s\n", books[i].title);
        printf("  Author: %s\n", books[i].author);
        printf("  Copies: %d\n", *books[i].copies);
    }
}


void updateCopies(struct Book *books, int n)
{
    char searchTitle[100];
    int found=0;
    printf("\nEnter the title of the book to update copies: ");
    scanf("%s",searchTitle);

    for (int i = 0; i < n; i++)
    {
        if (strcmp(books[i].title, searchTitle) == 0)
        {
            printf("\n");
            printf("Current copies: %d\n", *books[i].copies);
            printf("Enter new number of copies: ");
            scanf("%d", books[i].copies);
            printf("Updated copies: %d", *books[i].copies);
            printf("\n");
            found=1;
        }
    }
```

```
    if (found==0)
    {
        printf("Book not found");
        printf("\n");
    }

}

void freeMemory(struct Book *books, int n)
{
    for (int i = 0; i < n; i++)
    {
        free(books[i].title);
        free(books[i].author);
        free(books[i].copies);
    }
    free(books);
}
```

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.

```
#include <stdio.h>


struct Complex {
   float real;
   float imag;
};


struct Complex addComplex(struct Complex c1, struct Complex c2);
struct Complex multiplyComplex(struct Complex c1, struct Complex c2);
void displayComplex(struct Complex c);

int main() {
   struct Complex num1, num2, sum, product;
   char op;
   int a=0;


   printf("Enter the real and imaginary parts of the first complex number: ");
   scanf("%f %f", &num1.real, &num1.imag);


   printf("Enter the real and imaginary parts of the second complex number: ");
```

```
scanf("%f %f", &num2.real, &num2.imag);



while(a!=1)
{
   printf("Enter the operation (press e to exit): ");
   scanf(" %c",&op);
   switch(op)
   {
      case '+':
         sum = addComplex(num1, num2);
         printf("Sum: ");
         displayComplex(sum);
         break;


      case '*':
         product = multiplyComplex(num1, num2);
         printf("Product: ");
         displayComplex(product);
         break;


      case 'e':
         a=1;
         break;
```

```
        }
    }

    return 0;
}


struct Complex addComplex(struct Complex c1, struct Complex c2)
{
    struct Complex result;
    result.real = c1.real + c2.real;
    result.imag = c1.imag + c2.imag;
    return result;
}


struct Complex multiplyComplex(struct Complex c1, struct Complex c2)
{
    struct Complex result;
    result.real = c1.real * c2.real - c1.imag * c2.imag;
    result.imag = c1.real * c2.imag + c1.imag * c2.real;
    return result;
}


void displayComplex(struct Complex c) {
    if (c.imag >= 0)
        printf("%.2f + %.2fi\n", c.real, c.imag);
```

```
    else
        printf("%.2f - %.2fi\n", c.real, -c.imag);
}
```

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.

```
#include<stdio.h>

struct rectangle
{
    float length;
    float width;
};
float calculateArea(struct rectangle rect);
float calculatePerimeter(struct rectangle rect);
void displayResults(struct rectangle rect);

int main()
```

```
{
   struct rectangle num;
   char op;
   float area,perimeter;
   int a=0;
   printf("Enter the length of rectangle: ");
   scanf("%f",&num.length);
   printf("Enter the width of rectangle: ");
   scanf("%f",&num.width);

   while(a!=1)
   {
      printf("a- area p -pereimeter e- exit: ");
      scanf(" %c",&op);
      switch(op)
      {
         case 'a':
            area=calculateArea(num);
            printf("\nThe area is %f",area);
            printf("\n");
            break;

         case 'p':
            perimeter=calculatePerimeter( num);
            printf("\nThe perimeter is %f",perimeter);
            printf("\n");
            break;
```

```
        case 'e':
            a=1;
            break;
      }
   }

   return 0;
}



float calculateArea(struct rectangle num)
{
   return num.length * num.width;
}

float calculatePerimeter(struct rectangle num)
{
   return 2 * (num.length + num.width);
}
```

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.

```
#include <stdio.h>



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



void calculateGrade(struct Student *student);
void displayGrade(struct Student student);

int main()
{
   struct Student student;
   char op;
   int a = 0;



   printf("Enter the name of the student: ");
```

```
    scanf(" %s", student.name);

    printf("Enter the roll number of the student: ");
    scanf("%d", &student.roll_no);

    printf("Enter the marks in 5 subjects:\n");
    for (int i = 0; i < 5; i++)
    {

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



    calculateGrade(&student);

    displayGrade(student);

    return 0;
}


void calculateGrade(struct Student *student) {
    float total = 0, average;


    for (int i = 0; i < 5; i++)
    {
        total += student->marks[i];
```

```
    }

    average = total / 5;

    if (average >= 90)
    {
        student->grade = 'A';
    }
    else if (average >= 75)
    {
        student->grade = 'B';
    }
    else if (average >= 50)
    {
        student->grade = 'C';
    }
    else
    {
        student->grade = 'F';
    }
}

void displayGrade(struct Student student)
{
```

```
    printf("\nGrade: %c\n", student.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.

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

struct Point {
    float x;
    float y;
};

float calculateDistance(struct Point p1, struct Point p2);
int insidecircle(struct Point p, float radius);

int main()
```

```
{
    struct Point p1, p2;
    char op;
    float radius;
    int a= 0;


    printf("Enter the coordinates of the first point (x y): ");
    scanf("%f %f", &p1.x, &p1.y);


    printf("Enter the coordinates of the second point (x y): ");
    scanf("%f %f", &p2.x, &p2.y);


    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);


    while (a!=1)
    {
        printf("\nChoose an operation:\n");
        printf("d - Calculate distance between two points\n");
        printf("c - Check if a point is inside the circle\n");
        printf("e - Exit\n");
        printf("Enter your choice: ");
        scanf(" %c", &op);
```

```
switch (op)
{
   case 'd':
      {
         float distance = calculateDistance(p1, p2);
         printf("Distance between the points: %.2f\n", distance);
      }
      break;

   case 'c':
      {
         if (insidecircle(p1, radius))
            printf("Point p1 (%.2f, %.2f) is inside the circle.\n", p1.x, p1.y);
         else
            printf("Point p1 (%.2f, %.2f) is outside the circle.\n", p1.x, p1.y);



         if (insidecircle(p2, radius))
            printf("Point p2 (%.2f, %.2f) is inside the circle.\n", p2.x, p2.y);
         else
            printf("Point p2 (%.2f, %.2f) is outside the circle.\n", p2.x, p2.y);
      }
      break;

   case 'e':
      a= 1;
      break;
```

```
    }
  }

  return 0;
}



float calculateDistance(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 insidecircle(struct Point p, float radius)
{
  float distanceFromOrigin = sqrt(p.x * p.x + p.y * p.y);
  return distanceFromOrigin <= radius;
}
```

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.

```
#include <stdio.h>

struct Employee
{
   char name[50];
   int emp_id;
   float salary;
   float tax;
};


void calculateTax(struct Employee *employee);
void displayTax(struct Employee employee);

int main()
{
   struct Employee employee;


   printf("Enter the name of the employee: ");
   scanf("%s", employee.name);
```

```
    printf("Enter the employee ID: ");
    scanf("%d", &employee.emp_id);

    printf("Enter the salary of the employee: ");
    scanf("%f", &employee.salary);


    employee.tax = 0;


    calculateTax(&employee);


    displayTax(employee);

    return 0;
}


void calculateTax(struct Employee *employee)
{
    if (employee->salary < 50000)
    {
        employee->tax = employee->salary * 0.10;
    }
    else
    {
        employee->tax = employee->salary * 0.20;
```

```
    }
}


void displayTax(struct Employee employee)
{

    printf("Tax: $%.2f\n", employee.tax);
}
```

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 <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 services[10];
int service_count;
}vehicles[100];
int vehicle_count = 0;
void add();
void update();
void display();
void report();
int main() {
int op;
while (1) {
printf("\nVehicle Service\n");
printf("1. Add a new\n");
printf("2. Update service history\n");
printf("3. Display service details\n");
printf("4. Generate report\n");
printf("5. Exit\n");
```

```
printf("Enter your choice: ");
scanf("%d", &op);
switch (op) {
case 1:
add();
break;
case 2:
update();
break;
case 3:
display();
break;
case 4:
report();
break;
case 5:
printf("Exiting....\n");
return 0;
default:
printf("WRONG INPUT\n");
}
}
return 0;
}
void add() {
if (vehicle_count >= 100) {
printf("Error: Cannot add more vehicles. Capacity reached.\n");
return;
```

```
}
struct Vehicle new_vehicle;
printf("Enter vehicle license plate: ");
scanf("%s", new_vehicle.license_plate);
printf("Enter vehicle owner name: ");
scanf("%s", new_vehicle.owner_name);
printf("Enter vehicle type (car, bike, etc.): ");
scanf("%s", new_vehicle.vehicle_type);
new_vehicle.service_count = 0;
vehicles[vehicle_count++] = new_vehicle;
printf("Vehicle added successfully.\n");
}
void update() {
char license_plate[15];
printf("Enter the vehicle license plate to update service history:
");
scanf("%s", license_plate);
int found = 0;
for (int i = 0; i < vehicle_count; i++) {
if (strcmp(vehicles[i].license_plate, license_plate) == 0) {
found = 1;
if (vehicles[i].service_count >= 10) {
printf("Error: Maximum services reached for this
vehicle.\n");
return;
}
struct Service new_service;
printf("Enter service type (e.g., oil change, tire
```

```
replacement): ");
scanf("%s", new_service.service_type);
printf("Enter service cost: ");
scanf("%f", &new_service.cost);
printf("Enter service date (dd/mm/yyyy): ");
scanf("%s", new_service.service_date);
vehicles[i].services[vehicles[i].service_count++] =
new_service;
printf("Service history updated successfully.\n");
return;
}
}
if (!found) {
printf("Error: Vehicle with license plate %s not found.\n",
license_plate);
}
}
void display() {
char license_plate[15];
printf("Enter the vehicle license plate to display service details:
");
scanf("%s", license_plate);
int found = 0;
for (int i = 0; i < vehicle_count; i++) {
if (strcmp(vehicles[i].license_plate, license_plate) == 0) {
found = 1;
printf("\nService details for vehicle %s:\n",
vehicles[i].license_plate);
```

```
for (int j = 0; j < vehicles[i].service_count; j++) {
printf("Service %d: %s\n", j + 1,
vehicles[i].services[j].service_type);
printf("Cost: %.2f\n", vehicles[i].services[j].cost);
printf("Date: %s\n",
vehicles[i].services[j].service_date);
printf("----------------------------\n");
}
return;
}
}
if (!found) {
printf("Error: Vehicle with license plate %s not found.\n",
license_plate);
}
}
void report() {
float total_revenue = 0;
printf("\nSummary Report:\n");
printf("------------------------\n");
for (int i = 0; i < vehicle_count; i++) {
printf("Vehicle: %s, Owner: %s, Type: %s\n",
vehicles[i].license_plate, vehicles[i].owner_name,
vehicles[i].vehicle_type);
for (int j = 0; j < vehicles[i].service_count; j++) {
printf("Service %d: %s, Cost: %.2f, Date: %s\n", j + 1,
vehicles[i].services[j].service_type, vehicles[i].services[j].cost,
vehicles[i].services[j].service_date);
```

```
total_revenue += vehicles[i].services[j].cost;
}
printf("------------------------\n");
}
printf("Total revenue from all services: %.2f\n", total_revenue);
}
```