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;

};

void input(struct Student *student);

void display(const struct Student *student);

void freeMemory(struct Student *student);

int main() {

    int n;



    printf("Enter the no:of students: ");

    scanf("%d", &n);
```

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

    if (students == NULL) {

        printf("Allocation Error!\n");

        return 1;

    }

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

        printf("\nEnter details for student %d:\n", i + 1);

        input(&students[i]);

    }

    printf("\nStudent Details:\n");

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

        display(&students[i]);

    }

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

        freeMemory(&students[i]);

    }

    free(students);

    return 0;

}



void input(struct Student *student) {

    char tempName[100];



    printf("Enter roll number: ");
```

```
    scanf("%d", &student->roll_no);

    printf("Enter name: ");
    scanf("%s", tempName);

    student->name = (char *)malloc(strlen(tempName) + 1);
    if (student->name == NULL) {
        printf("Allocation Error!\n");
        exit(1);
    }
    strcpy(student->name, tempName);

    printf("Enter marks: ");
    scanf("%f", &student->marks);
}

void display(const struct Student *student) {
    printf("Roll No: %d\n", student->roll_no);
    printf("Name: %s\n", student->name);
    printf("Marks: %.2f\n", student->marks);
    printf("\n");
}
```

```c
void freeMemory(struct Student *student) {

    free(student->name);

}
```

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 {
    char *title;
    char *author;
    int *copies;
};
void inputBook(struct Book *book);
void displayBook(const struct Book *book);
void updateBook(struct Book *book);
void freeMemory(struct Book *book);

int main() {
    int n, choice, bookIndex;
```

```
    printf("Enter the number of books: ");
    scanf("%d", &n);
    struct Book *books = (struct Book *)malloc(n * sizeof(struct
Book));
    if (books == NULL) {
        printf("Allocation Error!\n");
        return 1;
    }
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for book %d:\n", i + 1);
        inputBookDetails(&books[i]);
    }
    printf("\nLibrary Books:\n");
    for (int i = 0; i < n; i++) {
        displayBookDetails(&books[i]);
    }
    printf("\nDo you want to update the number of copies for a book?
(1-Yes, 0-No): ");
    scanf("%d", &choice);
    if (choice == 1) {
        printf("Enter the index of the book to update (1 to %d): ", n);
        scanf("%d", &bookIndex);

        if (bookIndex >= 1 && bookIndex <= n) {
            updateBookCopies(&books[bookIndex - 1]);
        } else {
            printf("Invalid book index!\n");
        }
    }
    printf("\nUpdated Library Books:\n");
    for (int i = 0; i < n; i++) {
        displayBookDetails(&books[i]);
    }
    for (int i = 0; i < n; i++) {
        freeBookMemory(&books[i]);
    }
    free(books);

    return 0;
}
void inputBook(struct Book *book) {
    char tempTitle[100];
```

```
    char tempAuthor[100];
    int tempCopies;

    printf("Enter book title: ");
    scanf(" %s", tempTitle);
    book->title = (char *)malloc(strlen(tempTitle) + 1);
    if (book->title == NULL) {
        printf("Allocation Error!\n");
        exit(1);
    }
    strcpy(book->title, tempTitle);
    printf("Enter author name: ");
    scanf(" %s", tempAuthor);


    book->author = (char *)malloc(strlen(tempAuthor) + 1);
    if (book->author == NULL) {
        printf("Allocation Error!\n");
        exit(1);
    }
    strcpy(book->author, tempAuthor);
    printf("Enter number of copies: ");
    scanf("%d", &tempCopies);
    book->copies = (int *)malloc(sizeof(int));
    if (book->copies == NULL) {
        printf("Allocation Error!\n");
        exit(1);
    }
    *(book->copies) = tempCopies;
}
void displayBook(const struct Book *book) {
    printf("Title: %s\n", book->title);
    printf("Author: %s\n", book->author);
    printf("Copies: %d\n", *(book->copies));
    printf("\n");
}

void updateBook(struct Book *book) {
    int newCopies;
    printf("Enter new number of copies for '%s': ", book->title);
    scanf("%d", &newCopies);
    *(book->copies) = newCopies;
}
```

```c
void freeMemory(struct Book *book) {
    free(book->title);
    free(book->author);
    free(book->copies);
}
```

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;

}sum,product,a,b;

void add(struct Complex a, struct Complex b);

void mul(struct Complex a, struct Complex b);

int main() {



    printf("Enter the real & imaginary parts of the 1st complex number:
");

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

```c
    printf("Enter the real & imaginary parts of the 2nd complex number:
");
    scanf("%f %f", &b.real, &b.imag);



    add(a, b);
    mul(a,b);

    printf("\nSum: %.2f + %.2fi\n", sum.real, sum.imag);
    printf("Product: %.2f + %.2fi\n", product.real, product.imag);

    return 0;
}
void add(struct Complex a, struct Complex b) {
    sum.real =a.real + b.real;
    sum.imag = a.imag + b.imag;
}

void mul(struct Complex a, struct Complex b) {
    product.real = a.real * b.real - a.imag * b.imag;
    product.imag = a.real * b.imag + a.imag * b.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 l;

    float b;

}rect;

float area(struct Rectangle rect);

float perimeter(struct Rectangle rect);

int main() {

    float a, p;



    printf("\nEnter the length :: ");

    scanf("%f", &rect.l);

    printf("\nEnter the breadth:: ");

    scanf("%f", &rect.b);



    a= area(rect);
```

```
    p= perimeter(rect);



    printf("\nArea: %f\n", a);

    printf("Perimeter: %f\n", p);



    return 0;

}

float area(struct Rectangle rect) {

    return rect.l * rect.b;

}

float perimeter(struct Rectangle rect) {

    return 2 * (rect.l + rect.b);

}
```

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>

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

void grade(struct Student student);

int main() {
    printf("Enter student's name: ");
    scanf("%s", student.name);
    printf("Enter student's roll number: ");
    scanf("%d", &student.roll);

    printf("Enter marks in 5 subjects:\n");
    for (int i = 0; i < 5; i++) {
        printf("Subject %d: ", i + 1);
        scanf("%f", &student.marks[i]);
    }
    grade(student);

    return 0;
}

void grade(struct Student student) {
```

```c
    int sum = 0;
    int avg;

    for (int i = 0; i < 5; i++) {
        sum += student.marks[i];
    }
    avg = sum / 5;

    if (avg >= 90) {
        student.grade = 'A';
    } else if (avg >= 80) {
        student.grade = 'B';
    } else if (avg >= 70) {
        student.grade = 'C';
    } else if (avg >= 60) {
        student.grade = 'D';
    } else {
        student.grade = 'F';
    }
    printf("\nName: %s", student.name);
    printf("\nRoll Number: %d\n", student.roll);
    printf("Marks: ");
    for (int i = 0; i < 5; i++) {
        printf("%.2f ", student.marks[i]);
    }
```

```
    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;

}p1,p2;

float distance(struct Point p1, struct Point p2);

int incircle(struct Point p, float r);

int main() {

    float r;
```

```c
    printf("Enter Point 1 coordinates:: ");
    scanf("%f %f", &p1.x, &p1.y);

    printf("Enter Point 2 coordinates:: ");
    scanf("%f %f", &p2.x, &p2.y);

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

    float d = distance(p1, p2);
    printf("Distance between Point 1 & 2: %.2f\n", d);

    if (incircle(p1, r)) {
        printf("Point 1 is inside the circle.\n");
    } else {
        printf("Point 1 is outside the circle.\n");
    }

    if (incircle(p2, r)) {
        printf("Point 2 is inside the circle.\n");
    } else {
        printf("Point 2 is outside the circle.\n");
    }
```

```
    return 0;

}

float distance(struct Point p1, struct Point p2) {



    return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));

}



int incircle(struct Point p, float r) {



    float d = sqrt(pow(p.x, 2) + pow(p.y, 2));

    if (d <= r) {

        return 1;

    } else {

        return 0;

    }

}
```

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.

```c
#include <stdio.h>

struct Employee {

    char name[50];

    int emp_id;

    float salary;

    float tax;

}emp;

int salarytax(struct Employee emp);

int main() {

    printf("Enter employee name: ");

    scanf("%s", emp.name);

    printf("Enter employee ID: ");

    scanf("%d", &emp.emp_id);

    printf("Enter employee salary: ");

    scanf("%f", &emp.salary);



    emp.tax =salarytax(emp);



    printf("\nName: %s\n", emp.name);

    printf("Employee ID: %d\n", emp.emp_id);

    printf("Salary: %.2f\n", emp.salary);

    printf("Tax to be paid is: %.2f\n", emp.tax);
```

```
    return 0;

}

int salarytax(struct Employee emp) {

    float tax= 0;

    if (emp.salary < 50000) {

        tax = emp.salary * 0.10;

    }

    else {

        tax = emp.salary * 0.20;

    }

    return (int)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 centre 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.**

```c
#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);
}
```