ASSIGNMENT

```c
#include<stdio.h>

struct date
{
   int days;
   int months;
   int years;
};
int main(){
    struct date CurrentDate;
    struct date *ptr;
    ptr=&CurrentDate;
    (*ptr).days=22;
    (*ptr).months=11;
    (*ptr).years=2024;
    printf("Todays date is = %d-%d-%d",(*ptr).days,(*ptr).months,(*ptr).years);
    return 0;

}
```

Output
Todays date is = 22-11-2024

```c
#include<stdio.h>

struct date
{
   int days;
   int months;
   int years;
};
int main(){
    struct date CurrentDate;
    struct date *ptr;
    ptr=&CurrentDate;
    ptr->days=22;
    ptr->months=11;
    ptr->years=2024;
    printf("Todays date is = %d-%d-%d",ptr->days,ptr->months,ptr->years);
    return 0;

}
```

Output
Todays date is = 22-11-2024

```c
#include<stdio.h>
struct intPtrs{
    int *p1;
    int *p2;
};
int main(){
    struct intPtrs pointers;
    int i1=100,i2;
    pointers.p1=&i1;
    pointers.p2=&i2;

    *pointers.p2=180;
    printf("i1 = %d *pointers.p1 = %d\n",i1,*pointers.p1);
```

```
    printf("i2 = %d *pointers.p2 = %d\n",i2,*pointers.p2);
    return 0;
}
```

Output
i1 = 100 *pointers.p1 = 100
i2 = 180 *pointers.p2 = 180

```
#include<stdio.h>

struct names{
    char first[40];
    char last[40];
};
struct pNames{
    char *first;
    char *last;
};
int main(){
    struct names CAnames={"Abhinav","Karan"};
    struct pNames CPnames={"Abhinav","Karan"};

    printf("%s\t%s \n",CAnames.first,CPnames.first);
    printf("Size of CAnames = %d\n",sizeof(CAnames));
    printf("Size of CPnames = %d\n",sizeof(CPnames));
    return 0;
}
```

Output
Abhinav Abhinav
size of CAnames = 80
size of CPnames = 8

Structure as Arguments to Functions

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

struct names{
    char first[40];
    char last[40];
};
bool nameComparison(struct names,struct names);
int main(){
    struct names CAnames={"Abhinav","Karan"};
    struct names CPnames={"Abhinav","Karan"};
    bool b=nameComparison(CAnames,CPnames);
    printf("b = %d",b);
    return 0;
}
bool nameComparison(struct names CAnames,struct names CPnames){
    if(strcmp(CAnames.first,CPnames.first)==0){
        return true;
    }
    else{
        return false;
    }
}
```

Output
b = 1

Pointers to Structures as Arguments

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

struct names{
    char first[40];
    char last[40];
};
bool nameComparison(struct names *,struct names *);
int main(){
    struct names CAnames={"Abhinav","Karan"};
    struct names CPnames={"Abhinav","Karan"};
    struct names *ptr1,*ptr2;
    ptr1=&CAnames;
    ptr2=&CPnames;
    bool b=nameComparison(ptr1,ptr2);

//bool b=nameComparison(&CAnames,&CPnames);-->if we use this step instead of above ptr1,ptr2
                                            execution takes more fastly//

    printf("b = %d",b);
    return 0;
}
bool nameComparison(struct names *p1,struct names *p2){
    if(strcmp(p1->first,p2->first)==0){
        return true;
    }
    else{
        return false;
    }
}
```

Output
b = 1

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:
    1. int roll_no: Roll number
    2. char *name: Pointer to dynamically allocated memory for the student's name
    3. float marks: Marks obtained
2. Write a program to:

    1. Dynamically allocate memory for n students.
    2. Accept details of each student, dynamically allocating memory for their names.
    3. Display all student details.
    4. 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 n;
    struct Student *students;
    printf("Enter the number of students: ");
    scanf("%d", &n);
    students = (struct Student *)malloc(n * sizeof(struct Student));
    if (students == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
    for (int i = 0; i < n; i++) {
        printf("\nEntering details for student %d:\n", i + 1);
        printf("Enter roll number: ");
        scanf("%d", &students[i].roll_no);
        char temp[100];
        printf("Enter name: ");
        scanf(" %[^\n]s", temp);
        students[i].name = (char *)malloc((strlen(temp) + 1) * sizeof(char));
        if (students[i].name == NULL) {
            printf("Memory allocation for name failed.\n");
            return 1;
        }
        strcpy(students[i].name, temp);
        printf("Enter marks: ");
        scanf("%f", &students[i].marks);
    }
    printf("\nStudent Details:\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);
    }
    for (int i = 0; i < n; i++) {
        free(students[i].name);
    free(students);
    printf("\nAll allocated memory has been freed.\n");
    return 0;
}
```

Output

Enter the number of students: 2

Entering details for student 1:

Enter roll number: 1

Enter name: soumya

Enter marks: 45

Entering details for student 2:

Enter roll number: 2

Enter name: sneha

Enter marks: 50

Student Details:

Student 1:

Roll Number: 1

Name: soumya

Marks: 45.00

Student 2:

Roll Number: 2

Name: sneha

Marks: 50.00

All allocated memory has been freed.

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:
    1. char *title: Pointer to dynamically allocated memory for the book's title
    2. char *author: Pointer to dynamically allocated memory for the author's name
    3. int *copies: Pointer to the number of available copies (stored dynamically)
2. Write a program to:
    1. Dynamically allocate memory for n books.
    2. Accept and display book details.
    3. Update the number of copies of a specific book.
    4. Free all allocated memory before exiting.

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

struct Book {
    char *title;
    char *author;
    int *copies;
};
int main() {
    int n, choice, i;
```

```c
    struct Book *books;
    printf("Enter the number of books: ");
    scanf("%d", &n);
    books = (struct Book *)malloc(n * sizeof(struct Book));
    if (books == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
    for (i = 0; i < n; i++) {
        printf("\nEntering details for book %d:\n", i + 1);
        char temp[100];
        printf("Enter title: ");
        scanf(" %[^\n]s", temp);
        books[i].title = (char *)malloc((strlen(temp) + 1) * sizeof(char));
        if (books[i].title == NULL) {
            printf("Memory allocation for title failed.\n");
            return 1;
        }
        strcpy(books[i].title, temp);
        printf("Enter author: ");
        scanf(" %[^\n]s", temp);
        books[i].author = (char *)malloc((strlen(temp) + 1) * sizeof(char));
        if (books[i].author == NULL) {
            printf("Memory allocation for author failed.\n");
            return 1;
        }
        strcpy(books[i].author, temp);
        books[i].copies = (int *)malloc(sizeof(int));
        if (books[i].copies == NULL) {
            printf("Memory allocation for copies failed.\n");
            return 1;
        }
        printf("Enter number of copies: ");
        scanf("%d", books[i].copies);
    }
    printf("\nBook Details:\n");
    for (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);
    }
    printf("\nDo you want to update the number of copies for a specific book? (1 for Yes, 0
for No): ");
    scanf("%d", &choice);
    if (choice == 1) {
        char searchTitle[100];
        printf("Enter the title of the book to update: ");
        scanf(" %[^\n]s", searchTitle);
        int found = 0;
        for (i = 0; i < n; i++) {
            if (strcmp(books[i].title, searchTitle) == 0) {
                printf("Enter the new number of copies: ");
                scanf("%d", books[i].copies);
                printf("Updated successfully!\n");
                found = 1;
                break;
            }
        }
        if (!found) {
            printf("Book with title '%s' not found.\n", searchTitle);
```

```
        }
    }
    printf("\nUpdated Book Details:\n");
    for (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);
    }
    for (i = 0; i < n; i++) {
        free(books[i].title);
        free(books[i].author);
        free(books[i].copies);
    }
    free(books);
    printf("\nAll allocated memory has been freed.\n");
    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:
    1. float real: Real part of the complex number
    2. float imag: Imaginary part of the complex number
2. Write functions to:

    1. Add two complex numbers and return the result.
    2. Multiply two complex numbers and return the result.

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

```
#include<stdio.h>
#include<stdlib.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);
int main() {
    struct Complex n1, n2, sum,product;
    printf("Enter the First Complex Number with real and imaginary part: ");
    scanf("%f %f", &n1.real, &n1.imag);
    printf("Enter the Second Complex Number with real and imaginary part: ");
    scanf("%f %f", &n2.real, &n2.imag);
    sum = addComplex(n1, n2);
    printf("Sum of Complex Numbers is: %.2f + %.2fi\n", sum.real, sum.imag);
    product= multiplyComplex(n1,n2);
    printf("Product of Complex Numbers is : %2.f + %.2fi\n",product.real,product.imag);
    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 product;
    product.real = (c1.real * c2.real) - (c1.imag * c2.imag);
    product.imag = (c1.real * c2.imag) + (c1.imag * c2.real);
    return product;
}
```

Output

Enter the First Complex Number with real and imaginary part: 1

2

Enter the Second Complex Number with real and imaginary part: 3

2

Sum of Complex Numbers is: 4.00 + 4.00i

Product of Complex Numbers is : -1 + 8.00i

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:
    1. float length: Length of the rectangle
    2. float width: Width of the rectangle
2. Write functions to:
    1. Calculate and return the area of the rectangle.
    2. 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>
#include<stdlib.h>

struct Rectangle{
    float length;
    float width;
};
```

```c
float rectangleArea(struct Rectangle rect);
float rectanglePerimeter(struct Rectangle rect);
int main(){
    struct Rectangle rect;
    float area,perimeter;
    printf("Enter the Length of Rectangle :");
    scanf("%f",&rect.length);
    printf("Enter the Width of Rectangle :");
    scanf("%f",&rect.width);
    area=rectangleArea(rect);
    printf("Area of Rectangle = %.2f\n",area);
    perimeter=rectanglePerimeter(rect);
    printf("Perimeter of Rectangle = %.2f",perimeter);
}
float rectangleArea(struct Rectangle rect){
    return rect.length*rect.width;

}
float rectanglePerimeter(struct Rectangle rect){
    return 2*(rect.length+rect.width);
}
```

Output

Enter the Length of Rectangle :2

Enter the Width of Rectangle :5

Area of Rectangle = 10.00

Perimeter of Rectangle = 14.00

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:
    1. char name[50]: Name of the student
    2. int roll_no: Roll number
    3. float marks[5]: Marks in 5 subjects
    4. char grade: Grade assigned to the student
2. Write a function to:
    1. 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>
struct Student {
    char name[50];
    int roll_no;
    float marks[5];
    char grade;
};
void assignGrade(struct Student *student);
int main() {
    struct Student student;
    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 of the student in 5 subjects in 100:\n");
    for (int i = 0; i < 5; i++) {
        printf("Subject %d: ", i + 1);
        scanf("%f", &student.marks[i]);
    }
    assignGrade(&student);
    printf("\nStudent Name: %s", student.name);
    printf("Roll Number: %d\n", student.roll_no);
    printf("Marks: ");
    for (int i = 0; i < 5; i++) {
        printf("%.2f ", student.marks[i]);
    }
    printf("\nGrade: %c\n", student.grade);
    return 0;
}
void assignGrade(struct Student *student) {
    float total = 0;
    float 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 >= 60) {
        student->grade = 'C';
    }
    else if (average >= 50) {
        student->grade = 'D';
    }
    else {
        student->grade = 'F';
    }
}
```

Output:
Enter the name of the student: soumya
Enter the roll number of the student: 23
Enter the marks of the student in 5 subjects in 100:
Subject 1: 90
Subject 2: 99
Subject 3: 95

Subject 4: 92
Subject 5: 97

Student Name: soumyaRoll Number: 23
Marks: 90.00 99.00 95.00 92.00 97.00
Grade: A

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:
    1. float x: X-coordinate of the point
    2. float y: Y-coordinate of the point
2. Write functions to:
    1. Calculate the distance between two points.
    2. 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 {
    float x;
    float y;
};
float calculateDistance(struct Point p1, struct Point p2);
int isPointInsideCircle(struct Point p, struct Point center, float radius);
int main() {
    struct Point p1, p2, center;
    float radius;
    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 coordinates of the center of the circle (x y): ");
    scanf("%f %f", &center.x, &center.y);
    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);
    printf("Distance between points: %.2f\n", calculateDistance(p1, p2));
    printf("Point 1 is %s the circle\n", isPointInsideCircle(p1, center, radius) ? "inside" :
"outside");
    printf("Point 2 is %s the circle\n", isPointInsideCircle(p2, center, radius) ? "inside" :
"outside");
    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));
}
```

```c
int isPointInsideCircle(struct Point p, struct Point center, float radius) {
    float distanceFromCenter = calculateDistance(p, center);
    return distanceFromCenter <= radius;
}
```

Output

Enter the coordinates of the first point (x y): 1

2

Enter the coordinates of the second point (x y): 3

4

Enter the coordinates of the center of the circle (x y): 3

3

Enter the radius of the circle: 4

Distance between points: 2.83

Point 1 is inside the circle

Point 2 is inside the circle

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:
    1. char name[50]: Employee name
    2. int emp_id: Employee ID
    3. float salary: Employee salary
    4. float tax: Tax to be calculated (initialized to 0)
2. Write a function to:
    1. Calculate tax based on salary slabs (e.g., 10% for salaries below $50,000, 20% otherwise).
    2. 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;
};
```

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

int main() {
    struct Employee emp;
    printf("Enter Employee Name: ");
    scanf("%[^\n]", emp.name);
    getchar();
    printf("Enter Employee ID: ");
    scanf("%d", &emp.emp_id);
    printf("Enter Employee Salary: ");
    scanf("%f", &emp.salary);
    calculateTax(&emp);
    printf("Employee Details:\n");
    printf("Name: %s\n", emp.name);
    printf("Employee ID: %d\n", emp.emp_id);
    printf("Salary: $%.2f\n", emp.salary);
    printf("Tax: $%.2f\n", emp.tax);
    return 0;
}
```

Output

Enter Employee Name: soumya

Enter Employee ID: 23

Enter Employee Salary: 30000

Employee Details:

Name: soumya

Employee ID: 23

Salary: $30000.00

Tax: $3000.00

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:
    1. char license_plate[15]: Vehicle's license plate number
    2. char owner_name[50]: Owner's name
    3. char vehicle_type[20]: Type of vehicle (e.g., car, bike)
2. Define a nested structure Service inside Vehicle with fields:

    1. char service_type[30]: Type of service performed

2. float cost: Cost of the service
3. char service_date[12]: Date of service

3. Implement the following features:

1. Add a vehicle to the service center record.
2. Update the service history for a vehicle.
3. Display the service details of a specific vehicle.
4. Generate and display a summary report of all vehicles serviced, including total revenue.

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

typedef struct {
    char service_type[30];
    float cost;
    char service_date[12];
} Service;
typedef struct {
    char license_plate[15];
    char owner_name[50];
    char vehicle_type[20];
    Service service_history[10];
    int service_count;
} Vehicle;
Vehicle records[100];
int vehicle_count = 0;
void add_vehicle();
void update_service();
void display_service_details();
void generate_summary_report();
int main() {
    int choice = 0;
    while (choice != 5) {
        printf("\nVehicle Service Center Management\n");
        printf("1.Add a Vehicle\n2.Update Service History\n3.Display Service
Details\n4.Generate Summary Report\n5.Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                add_vehicle();
                break;
            case 2:
                update_service();
                break;
            case 3:
                display_service_details();
                break;
            case 4:
                generate_summary_report();
                break;
            case 5:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
```

```
        }
    }
    return 0;
}
void add_vehicle() {

    Vehicle new_vehicle;
    printf("Enter license plate: ");
    scanf(" %[^\n]", new_vehicle.license_plate);
    printf("Enter owner name: ");
    scanf(" %[^\n]", new_vehicle.owner_name);
    printf("Enter vehicle type: ");
    scanf(" %[^\n]", new_vehicle.vehicle_type);
    new_vehicle.service_count = 0;
    records[vehicle_count++] = new_vehicle;
    printf("....Vehicle added successfully....\n");
}
void update_service() {
    char license_plate[15];
    printf("Enter license plate of the vehicle: ");
    scanf(" %[^\n]", license_plate);
    for (int i = 0; i < vehicle_count; i++) {
        if (strcmp(records[i].license_plate, license_plate) == 0) {
            Service new_service;
            printf("Enter service type: ");
            scanf(" %[^\n]", new_service.service_type);
            printf("Enter service cost: ");
            scanf("%f", &new_service.cost);
            printf("Enter service date (DD/MM/YYYY): ");
            scanf(" %[^\n]", new_service.service_date);
            records[i].service_history[records[i].service_count++] = new_service;
            printf("Service record added successfully!\n");
            return;
        }
    }
    printf("Vehicle not found.\n");
}
void display_service_details() {
    char license_plate[15];
    printf("Enter license plate of the vehicle: ");
    scanf(" %[^\n]", license_plate);
    for (int i = 0; i < vehicle_count; i++) {
        if (strcmp(records[i].license_plate, license_plate) == 0) {
            printf("Owner Name: %s\n", records[i].owner_name);
            printf("Vehicle Type: %s\n", records[i].vehicle_type);
            printf("Service History:\n");
            for (int j = 0; j < records[i].service_count; j++) {
                printf("Service %d:\n", j + 1);
                printf("Type: %s\n", records[i].service_history[j].service_type);
                printf("Cost: %.2f\n", records[i].service_history[j].cost);
                printf("Date: %s\n", records[i].service_history[j].service_date);
            }
            return;
        }
    }
    printf("Vehicle not found.\n");
}
void generate_summary_report() {
    float total_revenue = 0;
    printf("Summary Report:\n");
    printf("License Plate   Owner Name  Vehicle Type  Total Services   Total Cost\n");
```

```
    for (int i = 0; i < vehicle_count; i++) {
        float total_cost = 0;
        for (int j = 0; j < records[i].service_count; j++) {
            total_cost += records[i].service_history[j].cost;
        }
        printf("%-15s %-12s %-13s %-15d %.2f\n",
               records[i].license_plate,
               records[i].owner_name,
               records[i].vehicle_type,
               records[i].service_count,
               total_cost);
        total_revenue += total_cost;
    }
    printf("Total Revenue: %.2f\n", total_revenue);
}
```