STRUCTURE AND POINTERS

```
#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("Today's date is %d/%d/%d",(*ptr).days,(*ptr).months,(*ptr).years);
}
```

Using -> operator

```
#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;
```

```
    ptr->days = 22;
    ptr->months = 11;
    ptr->years = 2024;

    printf("Today's date is %d/%d/%d",ptr->days,ptr->months,ptr->years);
}
```

Structure containing pointers

```
#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);
}
```

Character array and character pointer

```
#include<stdio.h>
struct names{
    char first[40];
    char last[40];
};
struct pNames{
    char *first;
```

```
    char *last;
};
int main(){
    struct names CAnames = {"Anjali","Lal"};
    struct pNames CPnames = {"Anjali","Lal"};

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

Output:

```
Anjali  Anjali
size of CAnames = 80
size of CPnames = 16
```

Structures as arguments to function

```
#include<stdio.h>
#include<string.h>
#include<stdbool.h>
struct names{
    char first[40];
    char last[40];
};
bool nameComparison(struct names CAnames,struct names CPnames);
int main(){
    struct names CAnames = {"Anjali","Lal"};
    struct names CPnames = {"Anjali","Lal"};

    bool b = nameComparison(CAnames,CPnames);
    printf("b = %d",b);
```

```
}
bool nameComparison(struct names CAnames,struct names CPnames){
   if(strcmp(CAnames.first,CPnames.first)==0){
      return true;
   }
   else{
      return false;
   }
}
```

Pointers to structures as function arguments

```
#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 = {"Anjali","Lal"};
   struct names CPnames = {"Anjali","Lal"};
   /*struct names *ptr1,*ptr2;
   ptr1 = &CAnames;
   ptr2 = &CPnames;*/

   bool b = nameComparison(&CAnames,&CPnames);
   printf("b = %d",b);

}
bool nameComparison(struct names *p1,struct names *p2){
   if(strcmp(p1->first,p2->first)==0){
```

```
    return true;
  }
  else{
    return false;
  }
}
```

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>
struct student{
  int roll_no;
  char *name;
  float marks;
};
int main(){
  struct student *students;
  int n;
  printf("Enter the number of students: ");
  scanf("%d",&n);
```

```
    students = (struct student *)malloc(n*sizeof(struct student));
    //printf("allocated");
    //struct student newstudents[n];
    for(int i=0;i<n;i++){
        students[i].name = (char *)malloc(100 * sizeof(char));
        if (students[i].name == NULL) {
            printf("Memory allocation for name failed\n");
            return -1;
        }
        printf("Enter the name of student: \n");
        scanf("%s",students[i].name);
        printf("Enter rol number: \n");
        scanf("%d",&students[i].roll_no);
        printf("Enter mark: \n");
        scanf("%f",&students[i].marks);
    }
    for(int i=0;i<n;i++){
        printf("Roll number: %d Name: %s mark=
%f\n",students[i].roll_no,students[i].name,students[i].marks);
    }

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

    free(students);

    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.

```
#include<stdio.h>
#include<stdlib.h>
struct Book{
   char *title;
   char *author;
   int *copies;
};
void updateCopies(struct Book *books, int n, char *title);
int main(){
   struct Book *books;
   int n;
   printf("Enter the number of books: \n");
   scanf("%d",&n);
   books = (struct Book *)malloc(n*sizeof(struct Book));
   for(int i=0;i<n;i++){
      books[i].title = (char *)malloc(100 * sizeof(char));
      books[i].author = (char *)malloc(100 * sizeof(char));
```

```
        printf("Enter book title: \n");
        scanf("%s",books[i].title);
        printf("Enter the name of author of the book: \n");
        scanf("%s",books[i].author);
        printf("Enter the number of available copies: \n");
        scanf("%d",&books[i].copies);
    }
    for(int i=0;i<n;i++){
        printf("Title of book: %s\nAuthor name: %s\nAvailable number of copies:
%d\n",books[i].title,books[i].author,books[i].copies);
    }
    char titleToUpdate[100];
    printf("Enter the title of the book to update the number of copies: \n");
    scanf(" %[^\n]", titleToUpdate);

    updateCopies(books, n, titleToUpdate);
    printf("\nUpdated book details:\n");
    for (int i = 0; i < n; i++) {
        printf("Title of book: %s\nAuthor name: %s\nAvailable number of copies: %d\n\n",
            books[i].title, books[i].author, books[i].copies);
    }


}
void updateCopies(struct Book *books, int n, char *title) {
    for (int i = 0; i < n; i++) {
        if (strcmp(books[i].title, title) == 0) {
            printf("Enter the new number of copies for the book '%s': \n", books[i].title);
            scanf("%d", &books[i].copies);
            printf("Updated number of copies for '%s' is now: %d\n", books[i].title, books[i].copies);
            return;
```

```
        }
    }
    printf("Book with title '%s' not found!\n", title);
}
```

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 add(struct Complex num1, struct Complex num2);
struct Complex multiply(struct Complex num1, struct Complex num2);
void print(struct Complex num) {
    if (num.imag < 0)
        printf("%.2f - %.2fi\n", num.real, -num.imag);
    else
        printf("%.2f + %.2fi\n", num.real, num.imag);
}
```

```
int main() {
    struct Complex num1, num2, sum, product;
    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);
    sum = add(num1, num2);
    product = multiply(num1, num2);
    printf("The sum is: ");
    print(sum);

    printf("The product is: ");
    print(product);

    return 0;
}
struct Complex add(struct Complex num1, struct Complex num2) {
    struct Complex result;
    result.real = num1.real + num2.real;
    result.imag = num1.imag + num2.imag;
    return result;
}
struct Complex multiply(struct Complex num1, struct Complex num2) {
    struct Complex result;
    result.real = (num1.real * num2.real) - (num1.imag * num2.imag);
    result.imag = (num1.real * num2.imag) + (num1.imag * num2.real);
    return result;
}
```

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;
};
int area(struct Rectangle dimensions);
int perimeter(struct Rectangle dimensions);
int main(){
   struct Rectangle dimensions;
   printf("Enter the length and width of rectangle: ");
   scanf("%f %f",&dimensions.length,&dimensions.width);
   float result1 = area(dimensions);
   printf("Area of rectangle: %.2f\n",result1);
   float result2 = perimeter(dimensions);
   printf("Perimeter of rectangle: %.2f",result2);

}
int area(struct Rectangle dimensions){
   float result1;
   result1= dimensions.length*dimensions.width;
   return result1;
}
int perimeter(struct Rectangle dimensions){
```

```
    float result2;
    result2 = 2*(dimensions.length+dimensions.width);
    return result2;
}
```

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 average(struct Student *pstudents,int n);
int main(){
    int n;
    printf("Enter the number of students: \n");
    scanf("%d",&n);
    struct Student students[n];
    for(int i=0;i<n;i++){
```

```
        printf("Enter the name of student: \n");
        scanf("%s",students[i].name);
        printf("Enter roll number: \n");
        scanf("%d",&students[i].roll_no);
        printf("Enter the marks of 5 subjects: \n");
        for(int j=0;j<5;j++){
            scanf("%f",&students[i].marks[j]);
        }
    }
    average(students,n);
    for(int i=0;i<n;i++){
        printf("Name: %s\n Roll number: %d\n",students[i].name,students[i].roll_no);
        printf("Marks: ");
        for(int j=0;j<5;j++){
            printf("%.2f ",students[i].marks[j]);
        }
        printf("\nGrade: %c\n",students[i].grade);
    }

    return 0;

}
void average(struct Student *pstudents,int n){
    for(int i=0;i<n;i++){
        float total = 0;
        for(int j=0;j<5;j++){
            total += (pstudents+i)->marks[j];
        }
        float avg = total/5.0;
        if(avg>=90){
            (pstudents+i)->grade='A';
```

```
    }
    else if(avg>=80){
      (pstudents+i)->grade='B';
    }
    else if(avg>=70){
      (pstudents+i)->grade='C';
    }
    else if(avg>=60){
      (pstudents+i)->grade='D';
    }
    else{
      (pstudents+i)->grade='F';
    }
  }
}
```

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;
};
void distance(struct Point point1,struct Point point2);
void check(struct Point point3,int r);
int main(){
    struct Point point1;
    struct Point point2;
    struct Point point3;
    printf("Enter the x and y coordinates of first point: ");
    scanf("%f %f",&point1.x,&point1.y);
    printf("Enter the x and y coordinates of first point: ");
    scanf("%f %f",&point2.x,&point2.y);
    distance(point1,point2);
    int radius;
    printf("\nEnter the radius of circle: \n");
    scanf("%d",&radius);
    printf("Enter the x and y coordinates of point to be checked: ");
    scanf("%f %f",&point3.x,&point3.y);
    check(point3,radius);



}
void distance(struct Point point1,struct Point point2){
    float d = sqrt((point2.x-point1.x)*(point2.x-point1.x)+(point2.y-point1.y)*(point2.y-point1.y));
    printf("Distance between the two points is %.2f",d);
}
void check(struct Point point3,int r){
    float sd = point3.x*point3.x+point3.y*point3.y;
    if(sd<r*r){
        printf("Point is inside the circle");
```

```
    }
    else if(sd>r*r){
        printf("Point is outside the circle");
    }
    else{
        printf("Point is on 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:
    - 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>
#include <string.h>
struct Employee {
    char name[50];
    int emp_id;
    float salary;
    float tax;
};
void calculateTax(struct Employee *emp)
```

```
int main() {
    struct Employee emp;
    printf("Enter employee name: ");
    fgets(emp.name, sizeof(emp.name), stdin);
    emp.name[strcspn(emp.name, "\n")] = 0;

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

    printf("Enter employee salary: ");
    scanf("%f", &emp.salary);
    emp.tax = 0.0;
    calculateTax(&emp);
    printf("\nEmployee Name: %s\n", emp.name);
    printf("Employee ID: %d\n",emp.emp_id);
    printf("Employee Salary: %.2f\n", emp.salary);
    printf("Calculated Tax: %.2f\n", emp.tax);

    return 0;
}
void calculateTax(struct Employee *emp) {
    if (emp->salary < 50000) {
        emp->tax = emp->salary * 0.10;
    } else {
        emp->tax = emp->salary * 0.20;
    }
}
```

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 <string.h>

#define MAX_VEHICLES 100
#define MAX_SERVICES 10
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[MAX_SERVICES];
    int service_count;
};
void addVehicle(struct Vehicle *vehicle);
void updateServiceHistory(struct Vehicle *vehicle);
void displayServiceDetails(struct Vehicle *vehicle);
void generateSummaryReport(struct Vehicle *vehicles, int vehicle_count);
int main() {
    struct Vehicle vehicles[MAX_VEHICLES];
    int vehicle_count = 0;

    int choice;

    while (1) {
        printf("1. Add a vehicle\n");
        printf("2. Update service history\n");
        printf("3. Display service details of a vehicle\n");
        printf("4. Generate summary report\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: {
                if (vehicle_count >= MAX_VEHICLES) {
                    printf("Cannot add more vehicles.\n");
                    break;
                }
                struct Vehicle new_vehicle;
                addVehicle(&new_vehicle);
                vehicles[vehicle_count++] = new_vehicle;
```

```
    printf("Vehicle added successfully.\n");
    break;
}
case 2: {
    char license_plate[15];
    printf("Enter the license plate number of the vehicle to update service history: ");
    scanf("%s", license_plate);

    int found = -1;
    for (int i = 0; i < vehicle_count; i++) {
        if (strcmp(vehicles[i].license_plate, license_plate) == 0) {
            found = i;
            break;
        }
    }

    if (found == -1) {
        printf("Vehicle with license plate %s not found.\n", license_plate);
        break;
    }

    updateServiceHistory(&vehicles[found]);
    break;
}
case 3: {
    char license_plate[15];
    printf("Enter the license plate number of the vehicle: ");
    scanf("%s", license_plate);

    int found = -1;
    for (int i = 0; i < vehicle_count; i++) {
```

```
            if (strcmp(vehicles[i].license_plate, license_plate) == 0) {
                found = i;
                break;
            }
        }

        if (found == -1) {
            printf("Vehicle with license plate %s not found.\n", license_plate);
            break;
        }

        displayServiceDetails(&vehicles[found]);
        break;
      }
      case 4:
        generateSummaryReport(vehicles, vehicle_count);
        break;
      case 5:
        printf("Exiting program.\n");
        return 0;
      default:
        printf("Invalid choice. Please try again.\n");
    }
  }

  return 0;
}
void addVehicle(struct Vehicle *vehicle) {
  printf("Enter license plate number: ");
  scanf("%s", vehicle->license_plate);
```

```
    printf("Enter owner's name: ");
    getchar();
    fgets(vehicle->owner_name, sizeof(vehicle->owner_name), stdin);
    vehicle->owner_name[strcspn(vehicle->owner_name, "\n")] = '\0';

    printf("Enter vehicle type (car, bike, etc.): ");
    fgets(vehicle->vehicle_type, sizeof(vehicle->vehicle_type), stdin);
    vehicle->vehicle_type[strcspn(vehicle->vehicle_type, "\n")] = '\0';

    vehicle->service_count = 0;
}

void updateServiceHistory(struct Vehicle *vehicle) {
    if (vehicle->service_count >= MAX_SERVICES) {
        printf("Cannot add more services. Maximum service limit reached.\n");
        return;
    }

    struct Service new_service;
    printf("Enter type of service performed: ");
    getchar();
    fgets(new_service.service_type, sizeof(new_service.service_type), stdin);
    new_service.service_type[strcspn(new_service.service_type, "\n")] = '\0';

    printf("Enter cost of the service: ");
    scanf("%f", &new_service.cost);

    printf("Enter service date (dd/mm/yyyy): ");
    scanf("%s", new_service.service_date);
    vehicle->services[vehicle->service_count++] = new_service;
    printf("Service history updated successfully.\n");
```

```
}
void displayServiceDetails(struct Vehicle *vehicle) {
    printf("\nService details for vehicle with license plate %s:\n", vehicle->license_plate);
    printf("Owner: %s\n", vehicle->owner_name);
    printf("Vehicle Type: %s\n", vehicle->vehicle_type);

    printf("Service History:\n");
    for (int i = 0; i < vehicle->service_count; i++) {
        printf("  Service Type: %s\n", vehicle->services[i].service_type);
        printf("  Service Cost: %.2f\n", vehicle->services[i].cost);
        printf("  Service Date: %s\n", vehicle->services[i].service_date);
    }
}
void generateSummaryReport(struct Vehicle *vehicles, int vehicle_count) {
    float total_revenue = 0.0;
    printf("\nSummary Report of All Vehicles Serviced:\n");
    printf("-------------------------------------------------\n");

    for (int i = 0; i < vehicle_count; i++) {
        printf("Vehicle %d:\n", i + 1);
        printf("  License Plate: %s\n", vehicles[i].license_plate);
        printf("  Owner: %s\n", vehicles[i].owner_name);
        printf("  Vehicle Type: %s\n", vehicles[i].vehicle_type);

        printf("  Service History:\n");
        for (int j = 0; j < vehicles[i].service_count; j++) {
            printf("    Service Type: %s\n", vehicles[i].services[j].service_type);
            printf("    Service Cost: %.2f\n", vehicles[i].services[j].cost);
            printf("    Service Date: %s\n", 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);
}
```