DAY_13_ASSIGNMENT

```
// Problem 1: Dynamic Student Record Management
// Objective: Manage student records using pointers to structures and dynamically allocate memory for student names.
// Description:
// 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
// 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 <string.h>
#include <stdlib.h>

struct student {
    int rollno;
    char *name;
    float marks;
};

int main() {
    struct student *class;
    int n;

    printf("Enter the number of students: ");
    scanf("%d", &n);
    class = (struct student *)malloc(n * sizeof(struct student));
    if (class == NULL) {
        printf("Memory allocation failed!\n");
        return 0;
    }
    for (int i = 0; i < n; i++) {
        char name1[100];

        printf("Enter the name of student %d: ", i + 1);
        scanf(" %[^\n]", name1);
        class[i].name = (char *)malloc((strlen(name1) + 1) * sizeof(char));
```

```
        if (class[i].name == NULL) {
            printf("Memory allocation failed for name!\n");
            return 1;
        }
        strcpy(class[i].name, name1);

        printf("Enter the roll number of student %d: ", i + 1);
        scanf("%d", &class[i].rollno);

        printf("Enter the marks obtained by student %d: ", i + 1);
        scanf("%f", &class[i].marks);
    }
    printf("\nStudent Details:\n");
    for (int i = 0; i < n; i++) {
        printf("Name: %s, Roll No: %d, Marks: %.2f\n", class[i].name, class[i].rollno, class[i].marks);
    }
    for (int i = 0; i < n; i++) {
        free(class[i].name);
    }
    free(class);

    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:
// 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)
// 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 libary{
    char *title;
    char *author;
    int *copies;
```

```
};

int main(){
    int n;
    printf("enter no of books:");
    scanf("%d",&n);

    struct libary * ptr=(struct libary *)malloc(n*sizeof(struct libary));

    for(int i=0;i<n;i++){
       ptr[i].title=(char*)malloc(100*sizeof(char));
        ptr[i].author=(char*)malloc(100*sizeof(char));
         ptr[i].copies=(int*)malloc(sizeof(int));
        printf("enter book name:");
        scanf("%s",ptr[i].title);
        printf("enter naame of the author :");
        scanf("%s",ptr[i].author);
        printf("enter no of copies:");
        scanf("%d",ptr[i].copies);
    }
    // display the details
        for(int i=0;i<n;i++){
        printf("%s\t%s\t%d\n",ptr[i].title,ptr[i].author,*ptr[i].copies);
        }
        // checking the copies
        for(int i=0;i<n;i++){
           char book[30];
           printf("enter name of the book to update copies");
            getchar();
           scanf("%[^\n]",book);

           if(strcmp(book,ptr[i].title)==0){
               (*ptr[i].copies)++;
               printf("Updated number of copies for '%s': %d\n", ptr[i].title, *ptr[i].copies);
           }

        }

        for(int i=0;i<n;i++){
            free(ptr[i].title);
            free(ptr[i].author);
            free(ptr[i].copies);
        }
```

```
    free(ptr);

}


// Problem 3: Student Grade Calculation
// Objective: Calculate and assign grades to students based on their marks by passing a
structure to a function.
// Description:
// 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
// Write a function to:
// Calculate the average marks and assign a grade (A, B, etc.) based on predefined criteria.
// Pass the structure by reference to the function and modify the grade field.

#include <stdio.h>

struct student {
    char name[50];
    int rollno;
    int marks[5];
    char grades;
};

void averagemarks(struct student *grades);

int main() {
    struct student grades;
    printf("Enter the name: ");
    scanf("%s", grades.name);
    printf("Enter the roll number: ");
    scanf("%d", &grades.rollno);
    printf("Enter the marks for 5 subjects:\n");
    for (int i = 0; i < 5; i++) {
        scanf("%d", &grades.marks[i]);
    }
    averagemarks(&grades);
    printf("\nStudent Details:\n");
    printf("Name: %s\n", grades.name);
    printf("Roll Number: %d\n", grades.rollno);
    printf("Marks: ");
```

```
    for (int i = 0; i < 5; i++) {
        printf("%d ", grades.marks[i]);
    }
    printf("\nGrade: %c\n", grades.grades);

    return 0;
}

void averagemarks(struct student *grades) {
    float total = 0.0;
    float average;
    for (int i = 0; i < 5; i++) {
        total += grades->marks[i];
    }
    average = total / 5;
    if (average >= 90) {
        grades->grades = 'A';
    } else if (average >= 75) {
        grades->grades = 'B';
    } else if (average >= 60) {
        grades->grades = 'C';
    } else if (average >= 50) {
        grades->grades = 'D';
    } else {
        grades->grades = 'F';
    }
}

// Problem 5: Employee Tax Calculation
// Objective: Calculate income tax for an employee based on their salary by passing a structure
to a function.
// Description:
// 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)
// 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.
// Pass the structure by reference to the function and display the updated tax in main.

#include<stdio.h>
struct salary {
```

```
    char name[50];
    int empid;
    float salary;
    float taxs;
};

void taxes(struct salary *tax);

int main() {
    struct salary tax;

    printf("Enter the employee name: ");
    scanf("%s", tax.name);
    printf("Enter the employee ID: ");
    scanf("%d", &tax.empid);
    printf("Enter the salary: ");
    scanf("%f", &tax.salary);
    taxes(&tax);
    printf("\nEmployee Details:\n");
    printf("Name: %s\n", tax.name);
    printf("Employee ID: %d\n", tax.empid);
    printf("Salary: %.2f\n", tax.salary);
    printf("Calculated Tax: %.2f\n", tax.taxs);
    return 0;
}
void taxes(struct salary *tax) {
    if (tax->salary > 50000) {
        tax->taxs = tax->salary * 0.20;
    } else {
        tax->taxs = tax->salary * 0.10;
    }
}


// Problem 1: Complex Number Operations
// Objective: Perform addition and multiplication of two complex numbers using structures
passed to functions.
// Description:
// Define a structure Complex with fields:
// float real: Real part of the complex number
// float imag: Imaginary part of the complex number
// Write functions to:
// Add two complex numbers and return the result.
// Multiply two complex numbers and return the result.
```

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

```
 #include<stdio.h>
struct complex{
   float real;
   float imag;
};
void multiplecomplex(struct complex *c1,struct complex *c2, struct complex *result);
void add_complex(struct complex *,struct complex *,struct complex *);
int main(){
   struct complex c1,c2,result;
   printf("enter real and imaginery part of first complex number\n");
   printf("enter value for real part:");
   scanf("%f",&c1.real);

   printf("enter value for imaginary part:");
   scanf("%f",&c1.imag);

   printf("enter real and imaginery part of second complex number\n");
   printf("enter value for real part:");
   scanf("%f",&c2.real);

   printf("enter value for imaginary part:");
   scanf("%f",&c2.imag);

   add_complex(&c1,&c2,&result);

  printf("sum of complex numbers=%.2f+%.2f",result.real,result.imag);
   multiplecomplex(&c1,&c2,&result);
   printf("sum of complex numbers=%.2f+%.2f",result.real,result.imag);



   return 0;
}

void add_complex(struct complex *c1,struct complex *c2,struct complex *result){
   result->real=c1->real+c2->real;
   result->imag=c1->imag+c2->imag;

}
void multiplecomplex(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 -c2->real*c1->imag;

}
```

```
// Problem 2: Rectangle Area and Perimeter Calculator
// Objective: Calculate the area and perimeter of a rectangle by passing a structure to functions.
// Description:
// Define a structure Rectangle with fields:
// float length: Length of the rectangle
// float width: Width of the rectangle
// Write functions to:
// Calculate and return the area of the rectangle.
// Calculate and return the perimeter of the rectangle.
// Pass the structure to these functions by value and display the results in main.
```

```
#include<stdio.h>


struct Rectangle {
    float length;
    float width;
};
float calculate_area(struct Rectangle r);
float calculate_perimeter(struct Rectangle r);

int main() {
    struct Rectangle r;
    printf("Enter the length of the rectangle: ");
    scanf("%f", &r.length);
    printf("Enter the width of the rectangle: ");
    scanf("%f", &r.width);
    float area = calculate_area(r);
    float perimeter = calculate_perimeter(r);
    printf("Area of the rectangle: %.2f\n", area);
```

```
    printf("Perimeter of the rectangle: %.2f\n", perimeter);

    return 0;
}
float calculate_area(struct Rectangle r) {
    return r.length * r.width;
}

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

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


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

struct point {
    float x;
    float y;
};

float distance(struct point p1, struct point p2);
void checkinside(struct point p, float radius);

int main() {
    struct point p1, p2, p;
    float radius;
    printf("Enter first point (x1, y1): ");
```

```
    scanf("%f %f", &p1.x, &p1.y);
    printf("Enter second point (x2, y2): ");
    scanf("%f %f", &p2.x, &p2.y);
    float x = distance(p1, p2);
    printf("Distance between two points: %.2f\n", x);
    printf("Enter radius: ");
    scanf("%f", &radius);
    printf("Enter points for test point (x, y): ");
    scanf("%f %f", &p.x, &p.y);
    checkinside(p, radius);
    return 0;
}

float distance(struct point p1, struct point p2) {
    return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));
}

void checkinside(struct point p, float radius) {
    float distance_from_origin = pow(p.x, 2) + pow(p.y, 2);
    float radius_squared = pow(radius, 2);

    if (distance_from_origin <= radius_squared) {
        printf("Point is inside the circle.\n");
    } else {
        printf("Point is not inside the circle.\n");
    }
}

// Problem Statement: Vehicle Service Center Management
// Objective: Build a system to manage vehicle servicing records using nested structures.
// Description:
// 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)
// 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
// 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>

struct Service{
    char service_type[30];
    float cost;
};
struct Vehicle{
    char license_plate[15];
    char owner_name[50];
    char vehicle_type[20];
    struct Service s[10];
};
void add(struct Vehicle v[],int *ptr);
void display(struct Vehicle v[],int *ptr);
void update(struct Vehicle v[],int *ptr);
void revenue(struct Vehicle v[],int *ptr);

int main(){
    struct Vehicle v[10];
    int ch,i=1;
    while(1){
        printf("\n\n1. Add an new record\n2.update service\n3. Display the detials\n4. Total
revenue\n5. Exit\n");
        scanf("%d",&ch);
        switch(ch){
            case 1: add(v,&i);
                    break;
            case 2: update(v,&i);
            case 3: display(v,&i);
                    break;
            case 4: revenue(v,&i);
                    break;
            // case 5: exit();
        }
    }
    // printf("Owner Name: ");
    // scanf("%s",v1.owner_name);
    // printf("License plate: ");
    // scanf("%s",v1.license_plate);
    // printf("Vechicle type: ");
    // scanf("%s",v1.vehicle_type);
    // printf("Service type: ");
    // scanf("%s",v1.s1.service_type);
```

```
    // printf("Cost amount: ");
    // scanf("%s",v1.s1.cost);

}
void add(struct Vehicle v[],int *ptr){
    printf("\n\nOwner Name: ");
    getchar();
    scanf("%[^\n]s",v[*ptr].owner_name);
    printf("License plate: ");
    getchar();
    scanf("%[^\n]s",v[*ptr].license_plate);
    printf("Vechicle type: ");
    getchar();
    scanf("%[^\n]s",v[*ptr].vehicle_type);
    printf("Service type: ");
    getchar();
    scanf("%[^\n]s",v[*ptr].s[*ptr].service_type);
    printf("Cost amount: ");
    getchar();
    scanf("%f",&v[*ptr].s[*ptr].cost);
    (*ptr)++;
}
void display(struct Vehicle v[],int *ptr){
    // int j=0;
    for(int j=1;j<*ptr;j++){
    printf("\n\nOwner Name: %s",v[j].owner_name);
    printf("\nLicense plate: %s",v[j].license_plate);
    printf("\nVechicle type: %s",v[j].vehicle_type);
    printf("\nService type: %s",v[j].s[j].service_type);
    printf("\nCost amount: %f",v[j].s[j].cost);
    }
}
void update(struct Vehicle v[],int *ptr){
    int ch,record;
    printf("Enter the record to be updated: ");
    scanf("%d",&record);
    if(record>*ptr){
        printf("Invalid record");
    }
    else{
        printf("1.Owner Name\n2.License plate\n3.Vechicle type\n4.Service type\n5.Cost update");
        scanf("%d",&ch);
        switch(ch){
            case 1: printf("\n\nOwner Name: ");
```

```
                getchar();
                scanf("%[^\n]s",v[record].owner_name);
                break;
        case 2: printf("License plate: ");
                getchar();
                scanf("%[^\n]s",v[record].license_plate);
                break;
        case 3: printf("Vechicle type: ");
                getchar();
                scanf("%[^\n]s",v[record].vehicle_type);
                printf("Cost amount: ");
                getchar();
                scanf("%f",&v[record].s[record].cost);
                break;
        case 4: printf("Service type: ");
                getchar();
                scanf("%[^\n]s",v[record].s[record].service_type);
                printf("Cost amount: ");
                getchar();
                scanf("%f",&v[record].s[record].cost);
                break;
        case 5: printf("Cost amount: ");
                getchar();
                scanf("%f",&v[record].s[record].cost);
                break;
      }
   }
}
void revenue(struct Vehicle v[],int *ptr){
   float total;
   for(int j=1;j<*ptr;j++){
      total+=v[j].s[j].cost;
   }
   printf("The Total Revenue of %d vechicles is %f",*ptr,total);
}
```

CLASS WORKS

```
#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);

     printf("Todays date is %d-%d-%d",(*ptr).days,(*ptr).months,(*ptr).years);
}
```

```
#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);
```

```
    //  printf("Todays 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 *pointer.p1=%d\n",i1,*pointers.p1);
    printf("i2=%d *pointer.p2=%d\n",i2,*pointers.p2);


}



// character array  and character pointer

#include<stdio.h>

struct names{
    char first[40];
    char second[40];
};
struct pName{
    char *first;
    char *last;
};
```

```
int main(){
    struct names CAnames = {"prasad","deepyhy"};
    struct pName Cpname = {"anusree","arya",};

    printf("%s\t%s \n",CAnames.first,Cpname.first);
    printf("size of cname =%d\n",sizeof(CAnames));
    printf("size of cpname =%d\n",sizeof(Cpname));

}
```

```
// structure  as argument to function

#include<stdio.h>
#include<string.h>
#include<stdbool.h>
struct names{
  char first[20];
  char last[20];
};


bool namecomparison(struct names ,struct names );


int main(){
    struct names Cnames={"nivya","nivya"};

    struct names Pnames={"navya","nivya"};

    bool b=namecomparison(Cnames,Pnames);

    printf("b=%d",b);

    return 0;
}


bool namecomparison(struct names Cnames,struct names Pnames){
```

```
    if(strcmp(Cnames.first,Pnames.first)==0){
        return true;
    }else{
        return false;
    }
}
```

```
#include<stdio.h>
#include<string.h>
#include<stdbool.h>
struct names{
  char first[20];
  char last[20];
};

bool namecomparison(struct names *,struct names * );

int main(){
    struct names Cnames={"nivya","nivya"};

    struct names Pnames={"navya","nivya"};

    struct names *ptr1,*ptr2;

    ptr1=&Cnames;

    ptr2=&Pnames;

    bool b=namecomparison(ptr1,ptr2);

    printf("b=%d",b);
```

```
    return 0;
}


bool namecomparison(struct names *ptr1,struct names *ptr2){
    if(strcmp(ptr1->first,ptr2->first)==0){
        return true;
    }else{
        return false;
    }
}


// Measure-Command { ./program_name.exe }
```