DAY !3

1)struct as a pointer

```
#include<stdio.h>

struct date{
  int days;
  int months;
  int year;

};

int main(){
   struct date currentDate;
   struct date *ptr;
   ptr=&currentDate;


   (*ptr).days=22;
   (*ptr).months=11;
   (*ptr).year=2024;

   printf("Todays Date =%d-%d-%d",(*ptr).days,(*ptr).months,(*ptr).year);




    return 0;
}
```

2)USING ARROW OPERATOR FOR STRUCT AS A POINTER

```
#include<stdio.h>

struct date{
  int days;
  int months;
  int year;

};

int main(){
   struct date currentDate;
   struct date *ptr;
   ptr=&currentDate;


   ptr->days=22;
   ptr->months=11;
    ptr->year=2024;

   printf("Todays Date =%d-%d-%d", ptr->days, ptr->months, ptr->year);




    return 0;
}
```

3)POINTER AS AN ELEMENT IN STRUCT

```
#include<stdio.h>
struct intptr{
    int *p1;
    int *p2;

};
int main(){

 struct intptr pointer;
 int i1=100,i2;
 pointer.p1=&i1;
 pointer.p2=&i2;
 *pointer.p2=98;



 printf("i1=%d\t p1=%d\n",i1,*pointer.p1);

  printf("i2=%d\t p2=%d",i2,*pointer.p2);



    return 0;
}
```

4) CHARACTER ARRAYS OR CHARACTER POINTERS

```
#include<stdio.h>

struct names{
  char first[20];
  char last[20];
};

struct pnames{
  char *first;
  char *last;
};

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

    struct pnames Pointernames={"navya","nivya"};

    printf("%s\t%s\n",Cnames.first,Cnames.last);
    printf("size of Cnames=%d",sizeof(Cnames));

    printf("size of Pnames=%d",sizeof(Pointernames));



    return 0;
}
```

5)
```
#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;
    }
}
```

6)POINTERS TO STRUCTURES AS FUNCTION ARGUMENTS

```
#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;
    }
}
```

7)**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(){
    int n;
    printf("enter no of students:");
    scanf("%d",&n);

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

    //printf("enter student details\n");

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

        ptr[i].name=(char*)malloc(100*sizeof(char));

        printf("enter student name:");
        scanf("%s",ptr[i].name);


        printf("enter roll no:");
        scanf("%d",&ptr[i].roll_no);

        printf("enter marks:");
        scanf("%f",&ptr[i].marks);

    }

    for(int i=0;i<n;i++){
        printf("%d\t%s\t%.2f\n",ptr[i].roll_no,ptr[i].name,ptr[i].marks);
    }

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

2)**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 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);



    }

        for(int i=0;i<n;i++){
        printf("Book Name:%s\tAuthor
Name:%s\tCopies:%d\n",ptr[i].title,ptr[i].author,*ptr[i].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);

}
```

1)**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;
};
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);


    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;

}
```

2)
/*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 area{
  float length;
  float width;

};
float area(struct area a1);
float perimeter(struct area p1);
int main(){
    struct area a1={5,6};
    struct area p1={2,3};

    float area1=area(a1);

    float perimeter1=perimeter(p1);
    printf("area of reactangle=%.2f\n",area1);
    printf("perimeter of reactangle=%.2f",perimeter1);



    return 0;

}

float area(struct area a1){
    float area_rectangle=a1.length*a1.width;
    return area_rectangle;


}

float perimeter(struct area p1){
    float perimeter_rect=2*(p1.length*p1.width);
    return perimeter_rect;
}
```

3)**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 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 coordinates of first point (x1, y1): ");
   scanf("%f %f", &p1.x, &p1.y);
```

```
    printf("Enter coordinates of 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 coordinates 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");
    }
```

```
}

5)#include<stdio.h>

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

void calculate_tax(struct employee *);


int main() {
   struct employee e1;
   struct employee *ptr = &e1;


   printf("Enter employee name: ");
   scanf("%[^\n]", ptr->name);
   getchar();

   printf("Enter employee ID: ");
   scanf("%d", &ptr->emp_id);

   printf("Enter salary: ");
   scanf("%f", &ptr->salary);


   calculate_tax(ptr);


   printf("\nEmployee Details:\n");
   printf("Name: %s\n", ptr->name);
   printf("Employee ID: %d\n", ptr->emp_id);
   printf("Salary: %.2f\n", ptr->salary);
```

```
    printf("Tax: %.2f\n", ptr->tax);

    return 0;
}
void calculate_tax(struct employee *ptr) {
     if (ptr->salary > 50000) {
        ptr->tax = ptr->salary * 0.20;
    } else {
       ptr->tax = ptr->salary  * 0.10;
    }
}
```

3)**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 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 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>

struct vehicle {
    char license_plate[15];
    char owner_name[50];
    char vehicle_type[20];
    struct service {
        char service_type[30];
        float cost;
        char service_date[12];
    } service;
};


void update_service(struct service *srv);
void add(struct vehicle *v1);
void display(struct vehicle v1);
void display_all(struct vehicle vehicles[], int count);

int main() {
    struct vehicle vehicles[10];
    int vehicle_count = 0;
    while (1) {
        int choice;
        printf("\nEnter choice (1-Add, 2-Update Service, 3-Display,
4-Display All, 0-Exit): ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                if (vehicle_count < 10) {
                    printf("ADD VEHICLE DETAILS\n");
                    add(&vehicles[vehicle_count]);
                    vehicle_count++;
                } else {
                    printf("Cannot add more vehicles. Maximum capacity
reached.\n");
                }
                break;

            case 2:
```

```
            printf("Enter vehicle index (0 to %d) to update service details: ",
vehicle_count - 1);
            int index;
            scanf("%d", &index);
            if (index >= 0 && index < vehicle_count) {
                update_service(&vehicles[index].service);
            } else {
                printf("Invalid vehicle index.\n");
            }
            break;

        case 3:
            printf("Enter vehicle index (0 to %d) to display details: ",
vehicle_count - 1);
            scanf("%d", &index);
            if (index >= 0 && index < vehicle_count) {
                display(vehicles[index]);  // Displays specific vehicle and
service details
            } else {
                printf("Invalid vehicle index.\n");
            }
            break;

        case 4:
            display_all(vehicles, vehicle_count);  // Displays all vehicles
and their service details
            break;

        case 0:
            printf("Exiting\n");
            return 0;

        default:
            printf("Invalid choice. Please try again.\n");
            break;
      }
   }
```

```
    return 0;
}


void add(struct vehicle *v1) {
    printf("Enter vehicle number: ");
    scanf("%s", v1->license_plate);

    printf("Enter owner name: ");
    scanf("%s", v1->owner_name);

    printf("Enter vehicle type: ");
    scanf("%s", v1->vehicle_type);

    printf("Enter service type: ");
    scanf("%s", v1->service.service_type);

    printf("Enter service cost: ");
    scanf("%f", &v1->service.cost);

    printf("Enter service date: ");
    scanf("%s", v1->service.service_date);
}


void update_service(struct service *srv) {
    printf("Enter new service type: ");
    scanf("%s", srv->service_type);

    printf("Enter new service cost: ");
    scanf("%f", &srv->cost);

    printf("Enter new service date (DD/MM/YYYY): ");
    scanf("%s", srv->service_date);
}
```

```
void display(struct vehicle v1) {
    printf("\n--- Vehicle Details ---\n");
    printf("License Plate: %s\n", v1.license_plate);
    printf("Owner Name: %s\n", v1.owner_name);
    printf("Vehicle Type: %s\n", v1.vehicle_type);
    printf("Service Type: %s\n", v1.service.service_type);
    printf("Service Cost: %.2f\n", v1.service.cost);
    printf("Service Date: %s\n", v1.service.service_date);
}


void display_all(struct vehicle vehicles[], int count) {
    printf("\n--- All Vehicle Details ---\n");
    for (int i = 0; i < count; i++) {
        printf("\nVehicle %d:\n", i + 1);
        display(vehicles[i]);
    }
}
```