1.// structures 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("todays date is = %d-%d-%d",(*ptr).days,(*ptr).months, (*ptr).years );

    return 0;
}
```

o/p

todays date is = 22-11-2024

2)//same prgm instead of using arrow 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;

    printf("todays date is = %d-%d-%d",ptr -> days,ptr -> months, ptr -> years );

    return 0;
}
```

o/p

todays date is = 22-11-2024

3)// 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 = -97;

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

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

    return 0;
}
```

o/p

i1 = 100   *pointers.p1 = 100
i2 = -97   *pointers.p2 = -97

4)// char arrays vs char pointers

```
#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;
}
```

o/p

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

5)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>

// Define the Student structure
struct Student {
    int roll_no;       // Roll number
    char *name;        // Pointer for dynamically allocated name
    float marks;       // Marks obtained
};

int main() {
    int n;
    printf("Enter the number of students: ");
    scanf("%d", &n);

    // Dynamically allocate memory for n students
    struct Student *students = (struct Student *)malloc(n * sizeof(struct Student));
    if (students == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }

    // Accept details of each student
    for (int i = 0; i < n; i++) {
        char temp[100]; // Temporary buffer for student name
```

```
    printf("\nEnter details for student %d:\n", i + 1);
    printf("Roll Number: ");
    scanf("%d", &students[i].roll_no);

    printf("Name: ");
    scanf(" %[^\n]", temp); // Read a string with spaces
    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("Marks: ");
    scanf("%f", &students[i].marks);
}

// Display all student details
printf("\nStudent Details:\n");
for (int i = 0; i < n; i++) {
    printf("Roll Number: %d\n", students[i].roll_no);
    printf("Name: %s\n", students[i].name);
    printf("Marks: %.2f\n", students[i].marks);
}

// Free all allocated memory
```

```
    for (int i = 0; i < n; i++) {
        free(students[i].name); // Free memory for name
    }
    free(students); // Free memory for students array

    printf("\nMemory freed successfully.\n");
    return 0;
}
```

6) 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<string.h>
#include<stdlib.h>
struct Book{
int *title;
```

```
char *author;

int copies;
};

int main(){

int n;

struct Book *book:

printf("enter the number of book \n");

scanf("%d",&n);

book = (struct Book *)malloc(n * sizeof(struct Book));

if(book==NULL) {

printf("memory allocation failed \n");
}
else
{

printf("memory allocated \n");
}
for(int i=0; i<n; i++)

{

printf("enter book details \n");

char temp[100];

printf("enter title\n");

scanf("%s",temp);

int l = strlen(temp)+1;

book[i].title=(char *)malloc(l*sizeof(char));

if(book[i].title==NULL)
{

printf("memory not allocated for name \n");
}
```

```
else{

printf("memory allocated for name \n");
}

strcpy(book[i].title,temp);

char auth[100];

printf("enter author\n");

scanf("%s", auth);

int m=strlen(auth)+1;

book[i].author=(char *)malloc(m*sizeof(char));

if(book[i].author==NULL)

{

printf("memory not allocated for name \n");

}

else

{

printf("memory allocated for name \n");

}

strcpy(book[i].author,temp);

printf("enter copies \n");

scanf("%d",&book[i].copies);

}

printf("book details \n");

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

{

printf("title:%s \n author: %s \n copies:%d \n", book[i].title, book[i].author, book[i].copies);
```

```
}

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

{

free(book[i].title);

free(book[i].author);

}

free(book);

Return 0;

}
```

7) 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:

0 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(struct complex, struct complex);
void multiply(struct complex, struct complex);
int main()
{
struct complex n1;
struct complex n2;
printf("number 1 \n");
printf("enter the real part: \n");
scanf("%f", &n1.real);
printf("enter the imaginery part \n");
scanf("%f",&n1.imag);
printf("number 2 \n");
printf("enter the real part: \n");
scanf("%f",&n2.real);
printf("enter the imaginery part \n");
scanf("%f",&n2.imag);
printf("addition of two complex numbers \n");
add(n1,n2);
printf("multiplication of two complex numbers \n");
multiply(n1, n2);
}
```

```
void add(struct complex n1, struct complex n2)

{

struct complex add;

add.real=n1.real+n2.real;

Add.imag = n1.imag + n2.imag;

printf("the sum is %.2f +%.2fi \n",add.real,add.imag);

}

void multiply(struct complex n1, struct complex n2)

{

struct complex multiply;

multiply.real=n1.real*n2.real;

multiply.imag-n1.imag*n2.imag;

printf("the sum is %.2f * %.2fi \n", multiply.real,multiply.imag);

}
```

8) 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;
};
float area(struct Rectangle);
float perimeter(struct Rectangle);
int main()
{
struct Rectangle rect;
printf("Enter the length of the rectangle: ");
scanf("%f", &rect.length);
printf("Enter the width of the rectangle: ");
scanf("%f", &rect.width);
printf("Calculating area...\n");
float a = area(rect);
printf("Area is %.2f\n", a);
printf("Calculating perimeter...\n");
float p = perimeter(rect);
printf("Perimeter is %.2f\n", p);
return 0;
```

```
}

float area(struct Rectangle rect) {

return rect.length * rect.width;

}

float perimeter(struct Rectangle rect) {

return 2* (rect.length + rect.width);

}
```

9) 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:

47%

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>

// Define the Student structure
struct Student {
```

```
    char name[50];
    int roll_no;
    float marks[5];
    char grade;
};

// Function to calculate average marks and assign grade
void calculateGrade(struct Student *s) {
    float total = 0.0, average;

    // Calculate the total marks
    for (int i = 0; i < 5; i++) {
        total += s->marks[i];
    }

    // Calculate the average
    average = total / 5;

    // Assign grade based on average
    if (average >= 90) {
        s->grade = 'A';
    } else if (average >= 75) {
        s->grade = 'B';
    } else if (average >= 60) {
        s->grade = 'C';
    } else if (average >= 50) {
        s->grade = 'D';
    } else {
        s->grade = 'F';
    }
}

int main() {
    struct Student student;

    // Input student details
    printf("Enter student's name: ");
    scanf("%s", student.name);
    printf("Enter roll number: ");
    scanf("%d", &student.roll_no);

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

    // Calculate grade by passing structure by reference
```

```
    calculateGrade(&student);

    // Display student details with grade
    printf("\nStudent Details:\n");
    printf("Name: %s\n", 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;
}
```

10) 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:

D 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, struct Point);
void circle(struct Point, float);
int main()
{
struct Point p1;
struct Point p2;
printf("point 1 \n");
printf("enter x \n");
scanf("%f", &p1.x);
printf("enter y \n");
scanf("%f",&p1.y);
printf("point 2 \n");
printf("enter x \n");
scanf("%f",&p2.x);
printf("enter y \n");
scanf("%f",&p2.y);
float de distance(p1,p2);
printf("distance between points is %.2f\n",d);
float r;
printf("enter the radius of circle \n");
scanf("%f",&r);
printf("point 1 \n");
```

```
circle(p1,r);

printf("point 2 \n");

circle(p2,r);

return 0;

}

float distance(struct Point p1, struct Point p2)

{

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

}

void circle(struct Point p,float r)

{

float o=sqrt(p.x* p.x + p.y * p.y);

if(o<=r)

{

printf("point inside circle \n");

}

else

}

printf("point not inside circle \n");

}

}
```

11)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>

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

// Function to calculate tax based on salary slabs
void calculateTax(struct Employee *e) {
    if (e->salary < 50000) {
        e->tax = e->salary * 0.10;  // 10% tax for salary below $50,000
    } else {
        e->tax = e->salary * 0.20;  // 20% tax for salary $50,000 and above
    }
}

int main() {
    struct Employee emp;

    // Input employee details
```

```
    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);

    // Initialize tax to 0
    emp.tax = 0;

    // Calculate tax by passing structure by reference
    calculateTax(&emp);

    // Display employee details with calculated tax
    printf("\nEmployee Details:\n");
    printf("Name: %s\n", emp.name);
    printf("Employee ID: %d\n", emp.emp_id);
    printf("Salary: $%.2f\n", emp.salary);
    printf("Calculated Tax: $%.2f\n", emp.tax);

    return 0;
}
```

o/p

```
Enter employee name: joe
Enter employee ID: 2
Enter employee salary: 2000

Employee Details:
Name: joe
Employee ID: 2
Salary: $2000.00
Calculated Tax: $200.00
```