DATE: LAB REPORT NO.: **8** SET: **A**
TITLE OF THE PROGRAM: **Nested Structure in C**

OBJECTIVES

- To understand the concept of nested structures in C.
- To learn how to define and access members of nested structures.
- To pass structures to functions.

REQUIREMENTS

1. C Compiler (e.g., GCC)
2. Computer System
3. IDE or Text Editor
4. OS compatible with the software

THEORY

In programming, a nested structure is a structure that contains another structure as a member. This allows the organization of complex data types within a single structure.

To define a nested structure in C, we use the following syntax:

```
struct outer_structure {
    type member1;
    type member2;
    struct inner_structure {
        type inner_member1;
        type inner_member2;
    } inner;
};
```

To access the members of a nested structure, we use the dot operator (.). For example, to access **inner_member1** in the nested structure above, we would use the following syntax:

```
struct outer_structure outer;
outer.inner.inner_member1 = value;
```

PROCEDURE (Program Code, Comment, and Output)

1. C Program to Implement Nested Structure

Program Code:

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

// Declaration of the main structure
struct Organisation {
   char organisation_name[20];
   char org_number[20];
// Declaration of the dependent structure
   struct Employee {
      int employee_id;
      char name[20];
      int salary;
 // variable is created which acts
 // as member to Organisation structure.
   } emp;
};

  int main()
  {
   struct Organisation org;
   strcpy(org.organisation_name, "KMCcollege");
   strcpy(org.org_number, "GFG1768");
   org.emp.employee_id = 101;
   strcpy(org.emp.name, "RAM");
   org.emp.salary = 400000;

   // Printing the details
   printf("Organisation Name : %s\n", org.organisation_name);
   printf("Organisation Number : %s\n", org.org_number);
   printf("Employee id : %d\n", org.emp.employee_id);
   printf("Employee name : %s\n", org.emp.name);
   printf("Employee Salary : %d\n", org.emp.salary);

   return 0;
}
```

Output

```
Organisation Name : KMCcollege
Organisation Number : GFG1768
Employee id : 101
Employee name : RAM
Employee Salary : 400000
```

Explanation:

- We declare a structure **Organisation** that contains another structure Employee as a member.
- We initialize the members of **Organisation** and Employee using strcpy and direct assignments.
- We print the details of the **organization** and the **employee**.

2. Passing Structure to a Function

```
#include <stdio.h>

struct student {

   char name[50];

   int age;

};

// function prototype

void display(struct student s);

int main()

{

  struct student s1;

   printf("Enter name: ");

   scanf("%[^\n]s", s1.name);

   printf("Enter age: ");
```

```
    scanf("%d", &s1.age);

    display(s1); // passing struct as an argument

    return 0;

}

void display(struct student s)

 {

    printf("\nDisplaying information\n");

    printf("Name: %s", s.name);

    printf("\nAge: %d", s.age);

}
```

Output

```
Enter name: John Doe
Enter age: 21

Displaying information
Name: John Doe
Age: 21
```

Explanation:

- We declare a structure **student** with two members: **name** and **age.**
- We define a function **display** that accepts a **student** structure as an argument and prints its members.
- In the main function, we read the **name** and **age** from the user, and pass the structure to the **display** function.

CONCLUSION

In this lab, we explored nested structures and their practical applications in C programming. We learned how to declare and access members of **nested structures**, and how to pass structures to functions. This knowledge is crucial for organizing complex data types and for efficient memory management in C programming.