Experiment No. 7

Familiarization with Structure

1. **Objective:** This experiment is intended to verify the concepts related to *structure* in the C programming language. Examples are set to help students fathom the concept meticulously.
2. **Theoretical Background:**

Structure is a user-defined datatype in C language which allows us to combine data of different types together. Structure helps to construct a complex data type which is more meaningful. It is somewhat similar to an Array, but an array holds data of similar type only. But structure on the other hand, can store data of any type, which is practical and more useful.

For example: If I have to write a program to store Student information, which will have Student's name, age, branch, permanent address, father's name etc, which included string values, integer values etc, how can I use arrays for this problem, I will require something which can hold data of different types together.
In structure, data is stored in the form of records.

Defining a structure
struct keyword is used to define a structure. struct defines a new data type which is a collection of primary and derived data types.

Syntax:

```
struct [structure_tag]
{
    //member variable 1
    //member variable 2
    //member variable 3
    ...
}[structure_variables];
```

As you can see in the syntax above, we start with the struct keyword, then it's optional to provide your structure a name, we suggest you to give it a name, then inside the curly braces, we have to mention all the member variables, which are nothing but normal C language variables of different types like int, float, array etc.
After the closing curly brace, we can specify one or more structure variables, again this is optional.
Note: The closing curly brace in the structure type declaration must be followed by a semicolon(;).

Example of Structure:

```
struct Student
{
    char name[25];
    int age;
    char branch[10];
    // F for female and M for male
    char gender;
};
```

Here struct Student declares a structure to hold the details of a student which consists of 4 data fields, namely name, age, branch and gender. These fields are called structure elements or members.
Each member can have different data types, like in this case, name is an array of char type and age is of int type etc. Student is the name of the structure and is called the structure tag.

Declaring Structure Variables

It is possible to declare variables of a structure, either along with structure definition or after the structure is defined. Structure variable declaration is similar to the declaration of any normal variable of any other datatype. Structure variables can be declared in following two ways:

1) Declaring Structure variables separately-

```
struct Student
{
    char name[25];
    int age;
    char branch[10];
    //F for female and M for male
    char gender;
};

struct Student S1, S2;      //declaring variables
of struct Student
```

2) Declaring Structure variables with structure definition-

```
struct Student
{
    char name[25];
    int age;
    char branch[10];
    //F for female and M for male
    char gender;
}S1, S2;
```

Here S1 and S2 are variables of structure Student. However this approach is not much recommended.

3. **Software requirements:**
    - Any IDE that supports a C compiler (preferably Code::Blocks; any other IDE or text editor with a C compiler installed will also be accepted.)

4. **Example Problems-**

Example 1: Define a structure “complex” to read two complex numbers and perform addition, subtraction of these two complex numbers and display the result.

Code:

```
#include<stdio.h>
struct complex{
 float real;
 float img;
};
int main(){
 struct complex c1,c2,add,sub;
  printf("Enter a and b of the first complex number (a+ib)
\n");
 scanf(" %f %f",&c1.real,&c1.img);
  printf("Enter a and b of the second complex number (a+ib)
\n");
 scanf(" %f %f",&c2.real,&c2.img);
 add.real = c1.real+c2.real;
 add.img = c1.img+c2.img;
 sub.real = c1.real-c2.real;
 sub.img = c1.img-c2.img;
 printf("c1+c2 = %f + %fi\n",add.real,add.img);
 if(sub.img<0) printf("c1-c2 = %f %fi",sub.real,sub.img);
 else printf("c1-c2 = %f + %fi",sub.real,sub.img);
 return 0;
}
```

Example 2: Write a menu driven program in 'C' which shows the working of the library. The menu option should be:
i) Add book details.
ii) Display book details.
iii) List all books of a given author.
iv) Show the count of books in the library.
v) Exit.
Create a structure called library to hold Book ID, title of the book, author name, price of the book.
Code:

```
#include<stdio.h>
#include<string.h>
struct library{
 int id;
 char title[40];
 char author[20];
 float price;
} b[100] ;
int num=0;
void Add(){
 printf("How many books' info do you want to enter? ");
 scanf(" %d",&num);
 for(int i=0;i<num;i++){
 printf("Enter the following information about the book:\n");
 printf("ID, title, author's name, price(in Tk)\n");
 scanf(" %d %s %s
%f",&b[i].id,&b[i].title,&b[i].author,&b[i].price);
 }
}
void Disp(){
 printf("\tID\tName\tAuthor\tPrice(Tk)\n");
 for(int i=0;i<num;i++){

printf("\t%d\t%s\t%s\t%f\n",b[i].id,b[i].title,b[i].author,b[i].p
rice);
 }
}
void Count(){
 printf("\nNo of books avalable in the library = %d\n",num);
}
void List(){
 char str[20];
 printf("Enter the author's name: ");
 scanf("%s",str);
 for(int i=0;i<num;i++){
 if(strcmp(str,b[i].author)==0)

printf("\n\t%d\t%s\t%s\t%f\n",b[i].id,b[i].title,b[i].author,b[i]
.price);
 }
}
int main(){
 int option=0;
 do {
 printf("\nWelcome to the library\nPlease Select an Option: \n");
printf("-----------------------------------------------------
--
\n");
 printf("1.Add book details\n2.Display book details\n3.List all
books of a given author\n4.Show total no. of books in the
library.\n5.Exit\n");

printf("-----------------------------------------------------
--
\n");
 scanf("%d",&option);
 switch(option){
 case 1: Add();
 break;
 case 2: Disp();
 break;
 case 3: List();
 break;
 case 4: Count();
 break;
 }
 }while(option != 5);
 return 0;
}
```

5. Lab Report:

Make the following modifications to the system in Example-2 -

- Add a password-protected authorization system so that option-1 can be used by the librarian only
- Enable the use of multi-word strings in book titles and author names
- Make the system dynamic: make sure that the newly input book info does not overwrite the previous ones