EENG 212
Structures

A Structure is a data type suitable for grouping data elements together. The elements or fields, which make up the structure use the four basic data types. As the storage requirements for a structure cannot be known by the compiler, a definition for the structure is first required. This allows the compiler to determine the storage allocation needed, and also identifies the various sub-fields of the structure.

Example: A structure for storing students' data can be defined as follows:

```
        struct   students {
                char name[25];
                long st_num ;
                float GPA;
                };
```

This declares a **NEW data type** called *students*. This is a definition to the compiler**.** It does not create any storage space and cannot be used as a variable. In essence, it is a new data type keyword, like **int** and **char**, and can now be used to create variables. Other data structures may be defined as consisting of the same composition as the *students* structure. Following line

```
        struct  students   new_student;
```

defines a variable called *new_student* to be of the same data type as that of the newly defined data type struct *students*.

There are 2 ways to access the elements of a structure:

1. Dot notation (.)

   e.g new_student.st_num=009010;

2. Arrow notation (->)

   e.g. new_student->GPA=3.70;

```
/* Program to illustrate a structure */
        #include <stdio.h>
        struct date {                          /* global definition of type date */
                int month, day, year;
        };
        main()
        {
         struct date  today;
         today.month = 10;
         today.day = 22;
         today.year = 2007;
         printf("Todays date is %d/%d/%d.\n", today.day, today.month, today.year );          }
```

INITIALIZING STRUCTURES

This is similar to the initialization of arrays the elements are simply listed inside a pair of braces, with each element separated by a comma. The structure declaration is preceded by the keyword *static*

```
        static struct date today = { 15,09,007 };
```

ARRAYS OF STRUCTURES
We can have an arrays of structures. The following statement creates an array called *birthdays* of the same data type as the structure *date* :

```
        struct date birthdays[5];
```

We can access the elements of the array birthdays using the following statements:

```
        birthdays[1].month = 09;
        birthdays[1].day   = 15;
        birthdays[1].year  = 2007;
        --birthdays[1].year;
```

STRUCTURES WHICH CONTAIN STRUCTURES
Structures can also contain structures. Consider where both a date and time structure are combined into a single structure called *date_time*, eg,

```
        struct date {
                int  month, day, year;
        };
        struct time {
                int  hours, mins, secs;
        };
        struct date_time {
                struct date sdate;
                struct time stime;
        };
```

This declares a structure whose elements consist of two other previously declared structures. Initialization could be done as follows,

```
        static struct date_time today = { { 22, 10, 2007 }, { 16,55,33 } };
```

which sets the *sdate* element of the structure *today* to the twenty second of October, 2007. The *stime* element of the structure is initialized to sixteen hours, fifty five minutes, thirty-three seconds. Each item within the structure can be referenced if desired, eg,

```
        ++today.stime.secs;
        if( today.stime.secs == 60 )
              ++today.stime.mins;
```

Lab Work

1. Study the following program and try to determine what its output is going to be. Then compile and run it to display the output on your screen.

```
        /* TIME.C  Program updates time by 1 second using functions */


        #include <stdio.h>
        struct time {
          int hour, minutes, seconds;
          };

          void time_update( struct time* );
          main()
          {
          struct time current_time;
          printf("Enter the time (hh:mm:ss):\n");
          scanf("%d:%d:%d", &current_time.hour,&current_time.minutes,&current_time.seconds);
          time_update (&current_time);
          printf("The new time is %02d:%02d:%02d\n",current_time.hour,current_time.minutes,
current_time.seconds);
          }

          /* function to update time by one second */
          void time_update(struct time *new_time )
          {
                    ++new_time->seconds;
                    if (new_time->seconds == 60) {
                              new_time->seconds = 0;
                              ++new_time->minutes;
                              if(new_time->minutes == 60) {
                                        new_time->minutes = 0;
                                        ++new_time->hour;
                                        if(new_time->hour == 24)
                                                  new_time->hour = 0;
                                        }
                              }
                    }
```

H.W.3 (due after one week)

Write a program with structures which contains information found in a library's card catalog. It should contain the author name, the publishing year (in form of: yyyy), the company name, and status. Use functions to fill and display the books enrolled in the structures. Assume that there are 10 books in the library. in the status part if the book publishing year is before 1985 the program should indicate that this book is a *reference* otherwise it will show it *available* to borrow. The first 3 variables (author, publishing year and company) will be entered by user and the last variable (status) is filled by the program according to publishing year.

Note:

1) Please Send your Homework in the following Emails, but remember ***who is your lab instructor***

emu.clab2@gmail.com *for* ***Pouya's*** *Student*
eenglab212@gmail.com *for* ***Mohamad'****s Student*

2) Subject of email Should include student Number + Homework Number
For Example: "St. 15000012 Homework #1"

3) Your homework should be saved with your student number and attached as notepad.
For Example 15000012.txt