Programming Exercises

10.1 Define a structure data type called **time_struct** containing three members integer **hour,** integer **minute** and integer **second**. Develop a program that would assign values to the individual members and display the time in the following form:

16:40:51

10.2 Modify the above program such that a function is used to input values to the members and another function to display the time.

10.3 Design a function **update** that would accept the data structure designed in Exercise 10.1 and increments time by one second and returns the new time. (If the increment results in 60 seconds, then the second member is set to zero and the minute member is incremented by one. Then, if the result is 60 minutes, the minute member is set to zero and the hour member is incremented by one. Finally when the hour becomes 24, it is set to zero.)

10.4 Define a structure data type named **date** containing three integer members **day**, **month** and **year.** Develop an interactive modular program to perform the following tasks;

- To read data into structure members by a function
- To validate the date entered by another function
- To print the date in the format

  April 29, 2002

by a third function.

The input data should be three integers like 29, 4, and 2002 corresponding to day, month and year. Examples of invalid data:

31, 4, 2002 – April has only 30 days
29, 2, 2002 – 2002 is not a leap year

10.5 Design a function **update** that accepts the **date** structure designed in Exercise 10.4 to increment the date by one day and return the new date. The following rules are applicable:

- If the date is the last day in a month, month should be incremented
- If it is the last day in December, the year should be incremented
- There are 29 days in February of a leap year

10.6 Modify the input function used in Exercise 10.4 such that it reads a value that represents the date in the form of a long integer, like 19450815 for the date 15-8-1945 (August 15, 1945) and assigns suitable values to the members **day, month** and **year.**

Use suitable algorithm to convert the long integer 19450815 into year, month and day.

10.7 Add a function called **nextdate** to the program designed in Exercise 10.4 to perform the following task;

- Accepts two arguments, one of the structure **data** containing the present date and the second an integer that represents the number of days to be added to the present date.

- Adds the days to the present date and returns the structure containing the next date correctly.

Note that the next date may be in the next month or even the next year.

10.8 Use the **date** structure defined in Exercise 10.4 to store two dates. Develop a function that will take these two dates as input and compares them.

- It returns 1, if the **date1** is earlier than **date2**
- It returns 0, if **date1** is later date

10.9 Define a structure to represent a vector (a series of integer values) and write a modular program to perform the following tasks:

- To create a vector
- To modify the value of a given element
- To multiply by a scalar value
- To display the vector in the form
  (10, 20, 30, ...)

10.10 Add a function to the program of Exercise 10.9 that accepts two vectors as input parameters and return the addition of two vectors.

10.11 Create two structures named **metric** and **British** which store the values of distances. The **metric** structure stores the values in metres and centimetres and the British structure stores the values in feet and inches. Write a program that reads values for the structure variables and adds values contained in one variable of **metric** to the contents of another variable of **British.** The program should display the result in the format of feet and inches or metres and centimetres as required.

10.12 Define a structure named **census** with the following three members:

- A character array city [ ] to store names
- A long integer to store population of the city
- A float member to store the literacy level

  Write a program to do the following:

- To read details for 5 cities randomly using an array variable
- To sort the list alphabetically
- To sort the list based on literacy level
- To sort the list based on population
- To display sorted lists

10.13 Define a structure that can describe an hotel. It should have members that include the name, address, grade, average room charge, and number of rooms.

Write functions to perform the following operations:

- To print out hotels of a given grade in order of charges
- To print out hotels with room charges less than a given value

10.14 Define a structure called **cricket** that will describe the following information:

player name
team name
batting average

Using **cricket,** declare an array **player** with 50 elements and write a program to read the information about all the 50 players and print a team-wise list containing names of players with their batting average.

10.15 Design a structure **student_record** to contain name, date of birth and total marks obtained. Use the **date** structure designed in Exercise 10.4 to represent the date of birth.

Develop a program to read data for 10 students in a class and list them rank-wise.