Practice Problem on Array of Structures. **Try to solve this problem within a time limit of 1.5 hours (90 minutes).**

TXTMSG (contributed by F. R. Salvador)

1. INTRODUCTION

Dual SIM cellular phones have two slots for two SIM cards. Each SIM card is associated with an **inbox**. Thus, when there are two SIM cards in a phone, there will be two "inboxes". In each inbox there are messages. Each message includes the cell number of the sender, the actual message, the date and time the message was received, and a flag that indicates whether the message has been read or not.

A possible data structure declaration for representing an inbox as a C structure is encoded in the header file named as **`inbox.h`**. Open and study the header file's contents.

A description of how data can be stored in the inbox is given bulleted list below[1]. Make sure that you understand each item.

- Message values (each with a data type of **`struct msgTag`**) are added/stored in the member **`Msg[]`** array starting from index 0. Thus, the 1st message is stored in **`Msg[0]`**, the 2nd message in **`Msg[1]`**, the 3rd message in **`Msg[2]`** and so on…
- The **`n_msgs`** member indicates the number of messages currently stored in the **Msg[]** array. For example, when the value of **`n_msgs`** is 3, it means that there are three messages which are stored in **`Msg[0]`**, **`Msg[1]`** and **`Msg[2]`**.
- In terms of date and time, the oldest message is stored in **`Msg[0]`**. The most recent message is therefore stored in **`Msg[n_msgs - 1]`**.

2. SKELETON FILE AND OTHER FILES

a. **`TXTMSG-LASTNAME.c`** - this is the skeleton file that you'll need to edit as base code. Make sure to read and understand the contents and instructions in the skeleton file. Don't forget to rename this file with your own last name. For example, if your last name is SANTOS, then the file should be renamed as TXTMSG-SANTOS.c.
b. **`TXTMSG-LASTNAME-main`**.c - this is the file that contains the **`main()`** function. Rename this file with your own last name.
c. **`inbox.h`** - this is a header file that contains the structure type declaration(s) for this problem.
d. **`txtmsgs.bin`** - this is a binary file that contains data used in the problem. You do NOT need to OPEN this file since you won't be able to make sense out it because, again, it's a binary file!
e. **`loader-txtmsg.c`** – this file contains **`Load_Data()`** function that reads/loads data from **`txtsmgs.bin`** file and stores it to an array of structure. Study the codes, but you really do NOT need to understand all the details for this problem. It involves binary file processing that you may not be familiar with at this point in time.

3. REQUIRED TASKS

Open and study the contents of the skeleton file **`TXTMSG-LASTNAME.c`** and **`TXTMSG-LASTNAME-main.c`**. Define five functions that return values as answers to the following questions:

Q1: Is inbox *<inbox_number>* full? The word "full" means that new messages cannot be stored anymore. The answer to this question is either 1 (meaning yes or true, i.e., full) or 0 (meaning no or false, i.e., not full).

Q2: Who sent the most recent message in inbox *<inbox_number>*? *(note: return the cel number of the sender as answer)*

Q3: How many unread messages are there in inbox *<inbox_number>*?

Q4: What is the string length of the most recent message in inbox *<inbox_number>*?

Q5: How many messages were received from *<sender>* on *<date>* in inbox *<inbox_number>*?

DO NOT CALL `printf()` and `scanf()` in any of the required five function definitions! Printing is done only in `main()`.

4. SAMPLE RUNS and TESTING

Four sample runs on the next page show the input test values, and corresponding results based on the **Inbox** data loaded in the primary memory by the **`Load_Data()`** function. The numbers after **A1:** to **A5:** denote the respective answers to questions **Q1** to **Q5**.

Your solution should produce the same output using the same input values.

[1] Note that the given description is not necessarily how cellular phones actually store messages.

Sample Run #1: Input inbox number (type 0 or 1 only): 0 A1: 0 A2: 10006 A3: 13 A4: 12 Input sender cellphone number: 10001 Input numeric month day year separated by spaces: 1 1 2001 A5: 2	**Sample Run #2:** Input inbox number (type 0 or 1 only): 0 A1: 0 A2: 10006 A3: 13 A4: 12 Input sender cellphone number: 10006 Input numeric month day year separated by spaces: 1 4 2001 A5: 1
Sample Run #3: Input inbox number (type 0 or 1 only): 1 A1: 1 A2: 20005 A3: 10 A4: 6 Input sender cellphone number: 20002 Input numeric month day year separated by spaces: 2 1 2001 A5: 5	**Sample Run #4:** Input inbox number (type 0 or 1 only): 1 A1: 1 A2: 20005 A3: 10 A4: 6 Input sender cellphone number: 20005 Input numeric month day year separated by spaces: 7 25 2019 A5: 0

5. SUBMIT YOUR FILES VIA CANVAS

Submit/upload two files before the Canvas deadline. Don't forget to rename your files with your own last name.

a. **`TXTMSG-LASTNAME.c`**
b. **`TXTMSG-LASTNAME-main.`**

Back-up your solution (files) by sending it as an email attachment to your DLSU email account. Do not delete that email until you have completed CCPROG2.

6. TESTING & SCORING:

- Your program will be tested with a different set of test data, and/or different **`main()`** function.
- Each correct function definition will be given **10 points** each. Thus, the maximum total score will be 50/50.
- A program that has a syntax/compilation error will be given a score of 0 out 50.
- The score for an incorrect implementation of a required function is 0. For example, if only the answers to questions Q1 and Q2 are correct, then the score will be 20/50.

終