Assignment 2

Basic practice questions

Write programs to solve given problems or to generate specified output / behavior

1. Make a menu based program which provides following options to user

   (a) Add record

   (b) Modify record

   (c) Delete record

   (d) Search record

   (e) List all records

   (f) Save records to file

   (g) Read records from file

   (h) Exit

   The record will consist of only three fields student ID (or Roll number), Full name and email address. (Note that full name can contains spaces). For modify, delete and search user will specify record with student ID which would be unique. Hence, the program should not allow adding/modifying records in manner which leads to two records having same student ID.

   You can use only buffered I/O to perform all operatings including console I/O in printing menu, taking input and file I/O in saving and reading records from file. The program should ask name of file, when user asks to read from or save to a file.

   The program should save records which contain spaces like in full name, and special characters (like @) present in email addresses. Hence choose format of saving which candle handle all these things properly.

   (Hint: Save and read entire structure at a time)

2. Write the above program using only un-buffered I/O to perform all operations. (That is you should not use even scanf() and printf() function calls).

Assignment Questions

Write programs to solve given problems or to generate specified output / behavior

1. Write the program mentioned in basic practice problem. Here you can use any type of I/O commands for console input/output. For storing and reading data back from file you have to use sqlite database.

   Note that user should have option of exiting from program without saving the changes to any file or database. Also after making some changes if user goes and reads another database file then all current data should be discarded and only records read from the file should be present in memory.

2. Add search options to above program where user can specify only part of student ID, Name or email address and all records which contain that pattern should be listed. For example, if there are two records as shown below

   Record 1 : 200899004
   : Saurabh Barjatiya
   : saurabh.barjatiya@iiit.ac.in

   Record 2 : 200099007
   : Hello World
   : hello.world@iiit.ac.in

   Then searching for either of these strings should list these both records '200', '@iiit.ac.in', '@', '9900'. On the other hand if user type 'abh' in search then first record must be listed and second record should not be listed. Basically it is complete text search and you should not ask user whether the matching has to be performed based on student ID or email address or full name. The program should search for the specified part of string in all fields.

   Constraint that no two students should have same ID still applies.

Advanced practice questions

Write programs to solve given problems or to generate specified output / behavior

1. Check all the above programs (including basic practice problems, if you have attempted them) for buffer overflow and SQL injection issues.

For checking buffer overflow whenever you are taking input from outside whether it is through console or file, you must make sure that the character array you are going to store it into, is at least as large as the input. Since there is no guarantee that user or file will not have strings of size greater than some number 'X'. You should read only first 'N' bytes from console / file if your array is of size 'N+1'.

For checking SQL injection attacks you should make sure that the input specified by user themselves do not contain SQL queries. For example if some body enters his or her full name as

```
--'; DROP TABLE students;
```

then it should not lead to actual deletion of table from database. Hence you have to sanitize input and escape special characters like (') and (").

Both of these checks are extremely important for any production application to pass security requirements. The practice of checking database inputs in important for all programming languages or back-end databases that you may use.