ADDRESS BOOK APPLICATION WITH SIMPLE INDEXING

Read all of the instructions carefully, if a point isn't **CLEAR**, ask and **CLARIFY** it!

Due:
11th May, 11:59 pm

Goal

In this programming assignment you are asked to implement an Address Book application with Simple Indexing.

Implementation Details & Requirements

The application should be able to store information for typical address book entries, such as Name, Surname, Address and additional Notes. It must have basic functionality of Add/Update/Delete/Find/List records. Possible user interface for the main menu can be as follows:

```
My Address Book Application

Main Menu
=====================
1. Add New Record
2. Update A Record
3. Delete A record
4. Find A Record
5. List Records
6. Exit

Enter your choice ==> _
```

In order to index the file, you must use simple indexing approach, as shown in Figure 1. In this approach the records are stored in data file in **entry-sequenced order**. It means that new records are appended to end of file as they are entered. For record structure you can use whatever method you prefer (i.e., fixed length records, record separators). Additionally, the primary key of the system will be name+surname. Duplicate keys are not allowed, so you must check the presence of the record prior to insertion.

Beside, a separate index file must be created with keysorting approach. It means that the index file only keeps primary key field and the pointer to the data file records. Index file is kept in sorted order, and **when the application is started the whole file is read into RAM memory. Then all operations are handled through index file in RAM.** When the user wants to exit, flush the whole index file into disk.

Deletion will simply mark a record to be deleted. Record will not be actually deleted from the data file. However it will actually be deleted from the index file.

Search operation is to be done only by the primary key.

"List records" option will prompt the user for one-character-long input and will list the results that are starting with this character.

By the way, if the application encounters a run-time error, at that time synchronization between index and data file may fail since last update on index file on RAM not flushed into disk. You should design and implement a solution to overcome this issue. A typical solution can be adding a dirty-flag into index file header. During the program execution, in case of add, update or delete operations index structure in the RAM will change. Subsequent to this change, dirt-flag in the index file must be set. Therefore following a successful flushing of index file, you should re-set the

dirty-flag. When you open the data file in next time, check this flag. If it is set it means the file is not closed properly so it requires reorganization (i.e., re-creation of the index file)

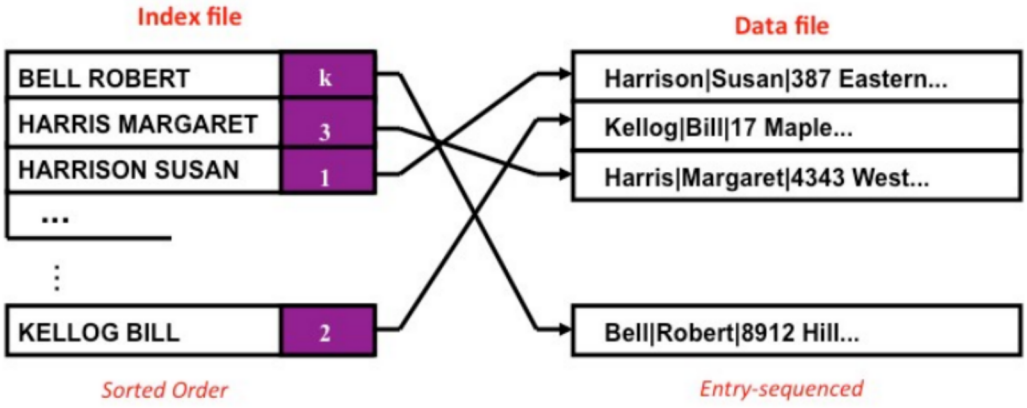

Figure-1: Simple Indexing structure

Documentation

Document your implementation neatly and clearly. Give as much detail as possible about your assumptions, how your program is supposed to work (no code).

Submission

Submission will be via Lecture.

- create makefile. (When make command is written on the console, the executable file named **contacts** will created)
- Put your source files (C files), makefile, data file and index file in a folder named as: StudentID_Name_Surname
- **You will loose credit for not naming your submission properly (10 points)**.
- You can submit your assignment **one day late**, but **you will loose 10 points**.

Honesty

Your submissions will be scanned among each other as well as the Internet repository. Any similarity sores over the threshold will directly get a zero. (No excuses). So we strongly encourage you not to submit your assignment rather than a dishonest submission.

Grading

Functionality(details will be shared)	80%
Coding Style	10%
Documentation	10%

For Questions

Mansur Tocoğlu are in charge of all questions related to this assignment. Contact him if you have any questions and he will try to answer any of your questions as soon as possible, except the ones "Hocam my program does not work, can you fix it" or "I have implemented it but it does not work, can you look at it". Debuggers are far more suitable options.