File Structure: Assignment #2 100 points



100

200

Notes:

- Cheaters will be graded by -ve points, Don't copy any code from anywhere ..
- Submit your code to through Acadox only. 2.
- Submit one compressed file with you ID and Group Name
- Due Date 17/4/2018 10:30 PM
- Team = max 2 students, team must be from the same lab

We want to store data about books and authors. (for simplicity consider any book has only one author, but any author may have more than one book, i.e. author has at least one book.)

Book attributes

Char [13]: Book ID //primary key Char [30]: Author_ID //sec. key

Char[50]: Book _Title Float: Book_Price

author attributes

Char [30]: Author_ID //primary key Char[50]: Author_Name //sec. key

Char[50]: Author_Address Char[11]: Author_Mobile

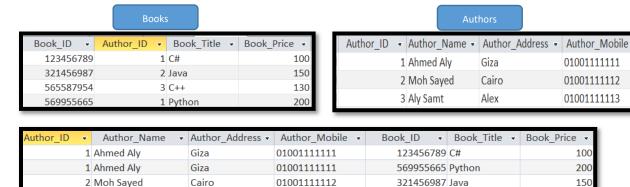
- Consider we want to save 10 books and 10 authors.
- Save the data for books and authors in the following format: delimited fields, length indicator records.
- You should develop the following indexes
 - Primary index using the Book_ID (for Books datafile)
 - Primary index using the Author_ID (for Authors datafile)

Alex

- Secondary index using Author_ID (for Books datafile) //Author_ID is sec. key in Books datafile
- Secondary index using Name (for Authors datafile)
- The user can write a query that contains fixed key words (formatted in red below)
- Examples for queries that user can write

3 Aly Samt

- **select all from Books where Author ID = 'xxxx'** // this query will use sec. index to get results
- select all from Authors where Author ID = 'xxxx' // this query will use primary, index to get results
- select Book Tile from Books where Book ID = 'xxxx' // this query will use sec. index to get results
- **select all from Books and Authors** // (check snapshot below as an example)



01001111112 321456987 Java 150 01001111113 565587954 C++ 130

select all from Books and Authors

File Structure: Assignment #2 100 points



the main welcome screen is below.

- 1) Add New Book
- 2) Add New Author
- 3) Delete Book (ID)
- 4) Delete Author (ID)
- 5) Print Book(ID)
- 6) Print Book(Author_ID)
- 7) Print Author(ID)
- 8) Print Author(name)
- 9) Write a Query
- 10) Exit

Important notes:

- All indexes are sorted ascending
- No need to use a status flag to check that indexes are up-to-date.
- But, you MUST implement secondary indexes using inverted list technique.
- Searching in indexes is performed using *binary search*.
- To delete a record just put an * in the beginning of that record. (no need for avail list implementation)
- All operations (add, delete) will affect indexes as explained in lecture 13.
- Search operations will use indexes (primary or secondary)
- Bind all secondary indexes with the primary index, don't bind them by addresses directly.

Assume any other information you need.