Overview

The goal of this project is to implement a program that parses an ASCII text-based CSV data file and then transforms that same information to a memory-efficient format and writes it to a binary data file. The new binary data file shall also have multiple index files for efficient record retrieval of**values** from fields other than the primary key. Your program should operate entirely from the command line (no GUI).

Requirements

- It is highly suggested to use Java for this project, but you may also use **C++**, **C#**, or **Python**.
- Your program should be named **MyDatabase**. The main method should therefore be contained in a file named **MyDatabase.java** (assuming you use Java).
- Your program should launch and then present the user with an text-based menu that provides the following options:
  - Parse the input CSV file to binary. Your program ***must*** store its binary database in a single file named "**data.db**". Your binary file***cannot*** use record separator charaters (e.g. newline, line feed, etc.), or field separators (e.g. commas, vertical bars (|), etc). Record location within the file must be stored in index files. Field location within record are known for fixed size data types and stored in a size byte for varchar.
  - Choose a field to query, then prompt for a value to compare, one of =, >, <, ≥, ≤, or ≠. If the input file has not yet been parsed to create your binary data file, a message should appear to prompt the user to first create the data file.
- Create one index file for each field. Index files should be named for the field(s) being indexed and have the extension .ndx, e.g. **id.ndx**,**company.ndx**.
  - Each index files should have one entry (one line) for each *unique* value in that field. Therefore, the id index (primary key) will have 1000 entries. Other fields may have fewer entires. The boolean fields will only have two entries (one for true, one for false).
  - For each entry in a given index file, a list of binary locations for all records that contain that value will follow.
  - Index file format may be either plain text (ASCII) or serialized binary objects, your choice. If you use plain text files for indexing, you***may*** use newlines to separate each entry.

Data Info

A CSV data file is attached: PHARMA_TRIALS_1000B.csv

Your program should transform the data into binary file where fields are converted to the binary types indicated below.

- The varchar field in the binary data file should be prefixed with a single byte that indicates the length of the string.
  - e.g. the string "GlaxoSmithKline LLC" should be converted to binary with the byte value **0x13** (i.e. "19", the length of the string) prefixed as a single byte.
  - Hexadecimal: 0x13 0x47 0x6c 0x61 0x78 0x6f 0x53 0x6d 0x69 0x74 0x68 0x4b 0x6c 0x69 0x6e 0x65 0x20 0x4c 0x4c 0x43
  - Chars: "<DC3>GlaxoSmithKline LLC", where <DC3> is the non-printable ASCII char 'device control 3'.
- Note that the four boolean fields should all be stored in the same byte. The first four most significant bits should always be false (i.e. 0000, hexadecimal 0x0). The next four bits should store the boolean values of the four fields respectively, in order. For example:

double_blind=**true**, controlled_study=**true**, govt_funded=**false**, and fda_approved=**true** would have the bits 00001101, which should be stored as the single byte**0x0d**.

- Therefore, record id=992 should take up a total **41 bytes** = 4 + (1 + 19) + 6 + 2 + 2 + 2 + 4 + 1 in the binary file.
  - It uses **70 bytes** (including the newline) in the CSV file).
  -

Field	Data Type	Binary Size
id	integer	4 bytes
company	varchar	variable (1 length byte + 1 byte/char)
drug_id	char(6)	6 bytes
trials	short int	2 bytes
patients	short int	2 bytes
dosage_mg	short int	2 bytes
reading	float	4 bytes
double_blind	boolean	1 byte
controlled_study	boolean	
govt_funded	boolean	
fda_approved	boolean	

Submission

- Submit your source code file(s), your binary data file, and your 11 index files as a zip'd or tar'd file with your NetID as a name, e.g. cid021000.zip, cid021000.gzip, cid021000.tar.
- Include a readme file that describes how to compile your code. Include language, version, and compile command string:
  - e.g. prompt> **javac MyDatabase.java**
- You do not need to submit the input CSV data file.

Q: Do we need to implement >=,>,<=,< for string and boolean data types.
A: No. Boolean fields only require equality test, i.e. "=="

Q: Is it mandatory to save indices in different files. I am saving indices in a map. My data structure is Map< String , Map < String , List > . Key —> column name , Value —> Map of column value and list of record pointers . For example : {id={1=[17165], 2=[18537]}, company={abc=[42,345,12],def=[344,44],……..} I am planning to serialize this map and save in one file. Is that ok?
A: If you are using serialized binary objects of data structures (e.g. an instance of a Java Collection, like the Map class) for indexes, you may save as a single binary index file. This is actually closer to the actual implementation of commercial databases. But this is outside of the scope of many CS-6360 students.