CS 301 Final Exam Project
Implement MLS Document Store
Fall 2021

Due Date – During the first 15 minutes of your scheduled final exam time – additional instructions will be posted. NO LATE submissions will be accepted.

Sample Data File - TBA
Sample Queries – TBA
Solutions to Queries - TBA

For this assignment you are to write a program to implement the operations required to process and execute NoSQL queries on a preexisting file of data. The data in the file corresponds to a collection in a database. In order to process the query, you will have to parse the query to identify which operations it is requesting, perform the operations on the specified documents and display the results. You must write the code to implement two operations similar to those in MongoDB: the find operation and the sort operation. You cannot run these queries using a database management system, instead you are implementing some of the software yourself for a NoSQL DBS. You may implement any sort algorithm you wish.
You must write your program in C.

Data: The data will be stored in a file named data.txt. In the data file each line represents a different document. Since this is a NoSQL database, the names of each field are included in the data along with the value of the field.

- The fields may be stored in a different order for each document. Each field is formatted as
  fieldname: value
- The field name is followed by a colon and a space, and fields are separated by a space.
- Assume all values for every field are integer.
- You should generate an ID field for each document. This system generated ID must be named A.
- Each document will have a security classification integer field called Y.
- All fields will be exactly one character long and will be one of the alphabetic characters from the capital letters B through W.

Example of possible input data for collection final:

```
B: 555 V: 1 C: 5 Y: 1
C: 10  V: 1 M: 555 Y: 2 B: 777 H: 20
M: 555 Y: 1 V: 2 C: 6
H: 20 V: 1 M: 555 B: 222 Y: 3
```

You can store the data anyway you choose for your collection. You are writing the DBMS, so do what you want to the data in order to process the query.

Queries: There will be a series of queries for you to process from an input file called final.txt. You must assume this as the name of the file. The NoSQL queries for this project are similar to MongoDB but they have a completely different syntax. You will implement the two operations FIND and SORT.

Operation - FIND

FIND: returns values to fields specified for documents that satisfy the specified condition(s).

- The query is formatted to appear on multiple lines.
- The first line of such input is a keyword FIND and optionally followed a security level (and preceded by a space).
- If no security level is specified, then all security levels are included in the query, otherwise, for a given security level, all the documents at a security classification <= the specified level are included in the query.
- The second line and subsequent lines of the input are the *conditions*.
- The final line for the query contains the *projections*, in other words, the fields to be displayed

Conditions: *fieldName ComparisonOp integer*

There will be zero or more select conditions with each select condition on a separate line.

- A document can only be included in the result if it satisfies **all** of the conditions. That means if a field that appears in a condition does not exist in the document, then that document does not appear in the result.
- If there are zero conditions then a line will contain a Z. Zero conditions means include all documents.
- The comparisonOp can only be one of these three: =, < or >.
- There will be spaces separating each fieldName, ComparisonOP and value. Examples are

  H = 67
  G = 7

Projections: *fieldName fieldName fieldName*

The list of fieldnames to be displayed appear on the last line of the query. The field names are separated by spaces.

- If all of the fields of the document are to be displayed, then the line will contain an X and only the letter X.
- The names of the fields can appear in any order.
- Unlike Mongo DB, the ID field, named A, is only included if specified. If a document does not have a field in the fieldName list, but it has other fields appearing in the list, then it should be included in the result.

The end of a query will be denoted with a ; preceded by a space. There will not be any blank lines in the final.txt or data.txt files.

```
FIND 2
H = 67
G > 7
A M K S G ;
FIND
Z
X ;
```

The last FIND query above returns the entire collection, e.g., all fields for all documents at all security levels, including the id, which has the fieldname of A.

Output: For each document satisfying the conditions, output each field name ending with a colon, a space, and the value for the field.

- Separate fields with a space.
- The fields should be displayed in the same order as they appear in the document, so the fields may not be in the same order as the fieldName list, similar to MongoDB.
- **Keep a count for each query executed and print the number of the query before you display the results**.

Example queries and their results:

```
    FIND 2
    M = 555
    C H ;
//Query 1
     C: 10 H: 20
     C: 6

    FIND 3
    H = 20
    M = 555
    X ;
//Query 2
     A: 2 C: 10  V: 1 M: 555 Y: 2 B: 777 H: 20
     A: 4  H: 20 V: 1 M: 555 B: 222 Y: 3

    FIND 1
    Y
    X ;
//Query 3
      A: 1 B: 555 V: 1 C: 5 Y: 1
      A: 3 M: 555 Y: 1 V: 2 C: 6
```

- If no documents satisfy the specified condition, output nothing.
- If the first line of the input for a query is not the keyword FIND or SORT print an error.
- Since this is a NoSQL DB, there should not be an error generated if there is no match for the name of a field. Instead:
    - If a field specified in a condition does not exist in the entire collection, it evaluates to a false so no output is produced.
    - If a field in a projection does not exist in a document, ignore it in the output. If other fields listed do exist, then list those in the output.

Example queries and their results:

```
     FIND
     T = 6
     U ;
```

//Query 4
//Returns nothing, if there is no T field and/or no U field

Operation - SORT
SORT: Returns the documents sorted on the specified fieldname for the specified security level. If no security level is specified, then all security levels are included.

- The query is formatted to appear on multiple lines.
- The first line of such input is a keyword SORT.
- The second line is the field on which the documents will be sorted.
- Sorting field is *fieldName = order*
- Where order is 1 for ascending and -1 for descending
- Only one field will be specified per SORT operation
- The entire document is displayed

Example: // Documents from the SORT below are listed in reverse sorted order of A for all security levels

```
    SORT
    A = -1 ;
//Query 5
    A: 4 H: 20 V: 1 M: 555 B: 222 Y: 3
    A: 3 M: 555 Y: 1 V: 2 C: 6
    A: 2 C: 10  V: 1 M: 555 Y: 2 B: 777 H: 20
    A: 1 B: 555 V: 1 C: 5 Y: 1
```

Example: // Documents listed in sorted order of field B for security classification 2

```
    SORT 2
    B = 1 ;
// Query 6
    A: 1 B: 555 V: 1 C: 5 Y: 1
    A: 2 C: 10  V: 1 M: 555 Y: 2 B: 777 H: 20
```

- If a document does not have the sort field, do NOT include it in the output. This is different from MongoDB which includes documents with missing sorting fields, but includes the missing sorting fields as { } in the output.
- If no documents contain the sort field, return nothing. Obviously, you cannot sort the documents on a field that does not exist.

NOTE: I am sure there may be questions about this assignment that I did not anticipate. Expect further clarifications to this assignment, so check it regularly. Start early so I can answer everyone's questions.