# *CS 210 Spring 2012*

# *Module 5 (Where Clause) Instructions*

1. Module 4 developed a mechanism for storing data in a binary file ("insert") and for reading it back out and printing to the command line ("read file_name"). In module 5, we will add to that functionality by implementing a "where clause".
2. The where clause consists of 3 components: a field name, a relational operator and a value. When confronted with a where clause, your program will need to examine all data in the binary file and determine which rows in the binary file contain a value in the field specified such that the resulting boolean is true; in short, the where clause acts as a filter on the rows in the table.
3. The where clause is potentially present in 3 of the commands in the grammar, and all three commands will be implemented in this module.
   a) Select – if the where clause is not present, "select" does the same thing as "print table_name", i.e., it prints out all rows in the table to the command line. If the where clause is present, it filters out a subset of rows and prints out only those rows.
   b) Update – update overwrites the designated field with the designated value as specified in the command. If there is no where clause, the field will be overwritten in all rows of the table. If the where clause is present, it will serve to filter which rows are to have the field updated.
   c) Delete – removes a row from further consideration. Without a where clause, all rows of the table are deleted. The where clause, if present, serves as a filter to determine which rows are deleted. How does one delete a row efficiently in a large table?
4. Javadocs will also be required unless you have received a Javadoc waiver.
5. Submissions should have all current code loaded into subversion. Submissions are made by emailing a working jar file labeled with your name to cs210.submissions@gmail.com, and should include all required class files, all source code for the application, and all javadocs as required.

HINTS:

1. Since you now have to skip around to read data in the binary file (such as reading one value from each row to test the where clause) it is helpful to add functionality to position your RAF pointer with a seek operation before any read or write. This can be calculated quite easily if you know the length of each row in the binary table (this is a handy value to store as a field in your Table class), the row number you are reading (which is actually reasonably easy to figure out) and the position of the field you are attempting to read within the row.
2. You can then read the appropriate single value from each row required by the where clause and pass it back to the Table class for processing. You will find polymorphism will be helpful in handling the values obtained, and if you have not already done so a set of "Type" or "Value" classes to hold values will be helpful.
3. There are 6 possible comparison operators in the where clause and 6 data types each of which have unique rules for comparison. To make this happen with manageable

complexity, it is helpful to implement the Comparable interface on the 6 datatype classes you create.

4. Then you should have all the tools necessary to set up the Where functionality which can be used whereever a where clause exists.
5. Please also note that the Delete command (removing one or more rows) does not lend itself to actually removing the row – this would require recopy of the binary file for each deleting and the overhead would be unacceptably high. Rather, a single byte in each row should be reserved as a “hidden” field and flagged on deletion.