Data Structures and Algorithms

Programming Assignment #2

Due: Friday February 21st

In this programming assignment, you will have to build a "mini-database" and give a user the capability of performing some simple queries.

The most common databases are so-called *relational databases* which are sets of *records*. Our database will be a class list with each record corresponding to a student enrolled in the class. Enrollment is limited by **Classsize = 100**. Each record has several fields called *attributes*. The records we will use will have 3 attributes:

```
StudentNumber          LastName          Grade
```

Attributes can be unique or non-unique. In our case, **StudentNumber**, an integer with 4 digits, is unique, i.e., no two students have the same number. The first digit of Student number indicates the academic age of the student, with 1 for Freshman, 2 for Sophomore, 3 for Junior, 4 for Senior and 8 for Graduate student.

On the other hand **Grade**, which will be an integer between 0 and 40, is non-unique since several students can have the same grade.

Finally, **LastName** in real life is not unique but will be unique for this assignment. LastName will be between 2 and 8 characters.

To provide rapid access to data, database systems use indexes. Each index allows the database to be searched (efficiently) on a selected attribute called the key of that index. In our example, Student-Number is the primary key, because it is unique, but indexes can also be built with other keys, unique or not, when fast access on those keys is desired.

In this assignment you will need to build the database and query it subsequently using 3 indexes (recall that the db has a maximum of 100 records):

- A binary search tree (BST) for StudentNumber (the BST need not be balanced), called *bstnum*.
- A hash table using linear probing for LastName, called *tbname*; Indicate in the README file, the size of the table and the hash function you used.
- A binary heap for Grade with the maximum value being at the root of the heap, named *heapgr*.

The first part of the assignment is to **Build** the database and the 3 indexes. The database will be given in the file *db.txt* with one record per line. Each record will have an integer (for StudentNumber), a string (for LastName), and another integer (for Grade) and a carriage return to end the line. For example, the first two lines of *db.txt* could be:
1234 JOE 35
4321 MARY 38
Once read in from *db.txt* the database should be stored in an Array of records, say db (Note, as said

earlier, that the array is limited to 100 entries). If the program were written in pseudocode, we could say something like:

```
db[0..99] : record array; //record has 3 fields
...
 ...if db[i].StudentNumber = 1234 then
         ...
```

Also, in the example above, JOE corresponds to the LastName field of db[0] and MARY corresponds to the LastName field of db[1]. (Tian will post programs in Java and C++ doing the kind of I/O that you will require.)

Once the array has been created, or while creating it (this is your choice), you should build the 3 indexes mentioned above. An entry in each of the indexes should have a field referring to the corresponding entry in the Array. For example, the unique node in *bstnum* with value "4321" should have a field indicating that the remainder of the record is stored in db[1].

After building the indexes, you should write methods to answer the following queries:

- List (by name) the students of academic age x (x = 1 or 2 or ...8) You should use the *bstnum* index for this method.
- List (by name) the students with the highest grade You should use the *heapgr* index for this method.
- What is the grade of Student xyz (you should also check that Student xyz is in the class roster). You should use the *tbname* index for this method.
- Any other interesting query you wish. This will count as a bonus and will be judged on correctness and difficulty.

These queries should be menu driven, i.e., once the database and indexes have been built, a menu of queries should appear on the screen and the user should be able to select one of the above queries as well as having the possibility to quit. (An example of such an interface can be found in the menu.java method of CSE 142 Autumn 2002 Homework 3.)

In addition to YOUR SUPERBLY DOCUMENTED PROGRAMS, you should include a README file indicating how to compile and run your program and whether we should be aware of something out of the ordinary.

(1) The main method must be in Dbq.java or Dbq.cpp.
(2) We will run your program only in command line:
java Dbq db.txt (or for C++: Dbq db.txt)

In summary, your program should build a database based on db.txt, build the 3 indexes, and then display a menu with the three required options and any additional query you wish.

You can use either Java or C++. You can use any source code given on the Web sites of the book (see CSE 373 "For more info" page). For additional information on java, the following link will be useful: http://java.sun.com/j2se/1.4.1/docs/api/

Instructions for Turn-in will be given later.