COMP 6521
Winter 2016
<u>Implementation Project</u>

Report Due: March 2, 2016
Demos: March 9, 2016

Summary:

In this project, you and your team are required to create a secondary, dense index on the AGE attribute of a text file PERSON which has about 20 billion records, each of which is 100 bytes and stored on a separate line.

NOTE: The file is sorted on the SIN number, which is the primary key. You are not allowed to reorder the records in this file, but rather scan the file and extract the information required to build the index.

Description:

The layout of the records in PERSON is as follows:

SIN	First Name	Last Name	Age	Yearly income	Address
9 Bytes	15 Bytes	15 Bytes	2 Bytes Integer	10 Bytes DEC (10.2)	**49 Bytes**

There is no special symbol used in the record to separate the attribute values. You will need to parse the data and extract the attribute value(s) that you need.

Using your index, we will be looking for people in a specific age, which a user will provide as an input to your program. The program produces two types of outputs: (1) the actual records of all such people in the file and (2) the number of such records returned.

Design and implement the best possible way to create and store the secondary index, and report technical details of your implementation, and for each input instance of the data file PERSON, report the number of disk I/O's used to produce the index, the time it took to produce the index, and the number of blocks used to store the index. For the index creation time and query processing time, start your clock when the program starts and stop it when the results are returned. See the next page for more details.

Running Your Program:

Consider the block size of 4KB. You will need to run your program twice with the following two different configurations, and compare and report the results.

Configuration 1:

Run your program with 5 MB of main memory available. Please see below for instructions on how to limit the main memory in the Eclipse IDE.

Configuration 2:

Run your program with 2 MB of main memory.

Data:

You will be provided with sample portions of PERSON to test your code, smaller than the file size mentioned above. Report the performance of your program on the whole file (20 billion records) which will be provided to you. On the demo day, we will use our instance of the data to evaluate your work.

The Report:

Your report should include at least the following sections:

- The group members: Full names and ID's
- Description of your program: The general structure of the code (main program and the sub-programs), the algorithm, how it accesses data, the architecture of your implementation indicating how the modules are connected and interact
- Description of your classes
- Difficulties faced: How they were addressed in your work, what difficulty(ies) were you unable to address. Limitations of your implementation. Note that these difficulties could be technical or personal
- Division of tasks: State who did what in the group.
- Results, analytics, and comments: average income for all age groups in the PERSON table provided, the number of disk I/O's, and time of execution for each configuration on a Concordia lab machine

Marking Scheme:

- Correct results 40%
- Results are returned in a reasonable amount of time 20%
- Code is well documented 10%
- Report is neat, clear, and answers all the questions 30%
- Bonus mark (5%) "good" increased speed-up for finding the records with the given age.

Limiting main memory in Eclipse IDE:

In Eclipse go to : *File -> Properties -> Run/Debug Settings*. Select the class which includes your main method, and then go to *(x) = arguments*.
In the program arguments, type the following line to set your main memory for 2 MB:

```
-Xms2m -Xmx2m
```

Apply and click *OK*.