CSI 2131 File Management
Computer Science

Winter 2006
University of Ottawa

Homework Assignment #2 (100 points, weight 10%)

Program 1 due: Friday, March 10 at 5:00 p.m. (via webCT)
Program 2 due: Friday, March 17 at 5:00 p.m. (via webCT)

Both programs deal with a datafile for student records that uses the following format: `Surname;Firstname;Student Number;Unit;Email`.
For example, file `toyexample.txt` can look like this:

```
Moura;Lucia Rosana;9871111;School of Management;LMOUR088@UOTTAWA.CA
Japkowicz;Nathalie;2925482;Faculty of Engineering;NAT@SITE.UOTTAWA.CA
Moura;Paulo;7657657;Faculty of Music;PMOUR706@UOTTAWA.CA
```

This file is composed of variable-length records with variable-length fields. Fields are separated by the character ";" and records are separated by newlines. Note that there is no ";" after the last field. You can get more familar with the file format by inspecting the input file provided.

1 Program 1: Building a primary index for a student file (50 marks)

In this part, your program will build a primary index for the given file for the primary key "Student Number". Then, the program will handle a sequence of searches by primary key, displaying the corresponding datafile record for each of them. You can expect that some searches will look for non-existing Student Numbers. Some further specifications and details are given next.

Implement your primary index as a simple index stored on a file.
The assumption is that the index would be too big to fit into an array in main memory. Instead, the index is organized as a fixed-length record file, with fixed-length fields: Student Number (primary key) with length 7 bytes (7 characters) and Reference Filed (Byte Offset of the corresponding record on the datafile) with length 4 bytes (1 long int). No newlines separate the records, so the record size is exactly 11 bytes. Remember that in a simple index, the keys (Student numbers) appear in increasing order on the file. The index file `toyexampleSNindex.txt` would look like:
`2925482xxxx7657657yyyy9871111zzzz`
where xxxx correspond to number 68, yyyy corresponds to number 138 and zzzz corresponds to number 0, with these numbers written in binary using 4 bytes each. Note that the student numbers appear in increasing order.

Build your index via successive additions into an empty index file
Implement a method `Addition` that performs a single addition to the index. Then, use the following steps to build the index:

1. Create an empty index file.
2. For each record in the datafile, read the datafile record, and insert its Student Number and ByteOffset into the primary index, by calling the method `Addition`.

Process a sequence of searches by student number using a file
An input file will be used to specify the searches, and will include a list of Student Numbers, separated by newlines. For each of these Student Numbers, do a binary search on the primary index file, retrieve its corresponding reference field, use it to locate the corresponding record on the datafile, and display the record contents (in its raw format) into the screen. If the binary search is not successfull to find this Student Number, output "RECORD NOT FOUND!". An example of such a file, `toysearches.txt`, and the output generate is given below:

```
---- toysearches.txt:-----
2925482
7676767
9871111
2925000
----- output -----------
Japkowicz;Nathalie;2925482;Faculty of Engineering;NAT@SITE.UOTTAWA.CA

RECORD NOT FOUND!

Moura;Lucia Rosana;9871111;School of Management;LMOUR088@UOTTAWA.CA

RECORD NOT FOUND!

-----------------------
```

Program input
You will prompt the user for datafile name, search file name, primary index file name.

1.1 Program 2: Secondary Index using Inverted Lists (50 marks)

In this program, you will be extending Program 1, by adding a secondary index for Last Name. This secondary index is a file organized as "inverted lists". Please, consult your class notes and/or the textbook for more details on how inverted lists are organized.
Your program will first build the primary and secondary index files, through a sequence of insertions to each index file. Then, your program will process a sequence of searches which can be by primary or by secondary key.

The Secondary Index is composed by the Secondary Key Index File and the Lists File
The Secondary Key Index File contains records of size 24 with two fields: Last Name (20 characters) and List Start (4 bytes). The Lists File contains records of size 11 with two

fields: Student Number (7 characters) and List Next (4 bytes).

Process a sequence of searches by Student Number or Last Name
The input file that specify the searches will include a letter indicating the type of search (S for Student Number and L for Last Name) followed by the key sought. An example of such file, `toymixsearches.txt`, and the output generated is given below:

```
---- toymixsearches.txt:-----
S2925482
S7676767
LMoura
S9871111
----- output -----------
Japkowicz;Nathalie;2925482;Faculty of Engineering;NAT@SITE.UOTTAWA.CA

RECORD NOT FOUND!

Moura;Paulo;7657657;Faculty of Music;PMOUR706@UOTTAWA.CA
Moura;Lucia Rosana;9871111;School of Management;LMOUR088@UOTTAWA.CA

Moura;Lucia Rosana;9871111;School of Management;LMOUR088@UOTTAWA.CA

------------------------
```

Program input
You will prompt the user for datafile name, search file name, primary index file name, secondary key index file name and secondary key lists file name.

IMPORTANT NOTE: Since the due date of program 2 is after program 1, you are allowed to use the solution provided for program 1 as the startup code for your program 2. This is not required, but it is advisable if you did not manage to make your program 1 work properly.

1.2 Submission instructions

By submitting this assignment, you are automatically acknowledging understanding of the policy on plagiarism available from the course web page.
EVERY FILE MUST CONTAIN A HEADER WITH Student Name, Student Number, and Course. Guidelines on style and documentation should be followed, as in assignment#1. Since you are writing longer programs, you should pay special attention to GOOD DESIGN and creation of classes to handle specific tasks.
Program 1: due March 10, 5:00 p.m.
Create a directory/folder named `a2part1` containing the source files for your Program 1 (include all .cpp and .h files that you create). Zip the folder `a2part1` and submit the zipped file as your assignment#2-part 1 submission to webCT (only one file is submitted).
Program 2: due March 17, 5:00 p.m. Do a similar submission for folder `a2part2`.