CSI 2131 File Management | Winter 2005
Computer Science | University of Ottawa

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

Due: Friday, March 4th at 5:00 p.m. (via webCT)

1 Datafile Design (30 marks)

In this part, you are going to design the field and record structure for a datafile. The datafile should have the following fields:

- semester (Fall, Winter or Summer)
- year (for example: 1999, 2003)
- course code (3 letters and 4 numbers, for example: CSI2131, CEG4185)
- course section (one letter such as A,B,C,D,P,W,etc. or empty)
- course title
- instructor's name
- instructor's home page (optional)
- course web page (optional)

The data will be read from a file containing each field on a different line. The given file wastes too much space with newlines and makes no attempt at storing the information more compactly.

In your datafile design, you are free to use any combination of the field/record structures seen in class, to choose how to store each field, to use compact notation, etc. Your datafile design should be as compact as possible, while still keeping all needed information and using reasonable assumptions about the generality of the file. For instance, it is reasonable to assume that new courses can be created, but it is OK to restrict the course codes to the existing programs (CSI, SEG, ELG, CEG, GNG).

Your mark will depend on the quality of the design and how compactly you stored the file.

You will write a program that reads data from the given file and writes it to the datafile in the format you designed. You will not hand-in the program, but you will hand in the datafile created. For this part you should hand in the following:

1. A report of no more than 1 page in which you describe the field and record structure used, the assumptions you made for your design, the number of bytes used for each field in a typical record (show an example and give sizes), and the total number of bytes used for the datafile.

2. The datafile you created containing the data from the file we gave you. This will be used in the program in the next part.

2 Program: Simple index supporting Insertion and Search (70 marks)

In this part, you will build a simple primary index for the datafile. This index will be stored in main memory (array sorted by the key) and it will be organized as described in class, that is, an array sorted by key and containing fields: primary key and reference field. The primary key will be the combination of the following fields: course code, term (Fall, Winter or Summer) and year. The reference field is the byte offset of the corresponding record in the datafile.
Your program will do the following:

1. Do a scan of the datafile (format you created) and create a simple index in main memory based on the primary key (course code + term + year).

2. Read from an input file (format given by us) a sequence of operations to perform in the datafile. Each operations may be of the following types:

   - Search (code "S"): given a primary key value, your program should search the file (via the index) for a record matching the given key value, and display its contents (to the screen) in a form readable by a human being.
   - Insert (code "I"): given all of the field values for a record, append the record to the datafile and make the corresponding insertion to the index.

   After every search, wait for the user to press a key in the keyboard before processing the next operation, so that the user can see the results of your search and any error messages for either operation.

You are free to choose the design for your classes. Here, we simply give you a suggestion to guide you in organizing your code. Suggested classes/elements of your program:

- datafile class (`datafile.h` and `datafile.cpp`): a class that manipulates the datafile, having methods for tasks like appending a record to the datafile and reading a record starting on a specified byte offset. Basic record manipulation could be done directly by this class or by a separate class used inside this one.

- index class (`index.h` and `index.cpp`): a class that contains the index, allowing methods for insertion of an index entry (primary key+ byte offset) and search (given primary key, retrieves the byte offset). Note that the index class doesn't have to know about the datafile class and vice-versa. Both classes may be jointly handled by the main program or by a third class of your choice.

- main program: declares required objects, controls the index creation through calls to methods from appropriate classes, and reads the required operations making calls to methods that process them. A general suggestion is that if your main program becomes too big, you should think of moving certain tasks or functionality to classes instead.

IMPORTANT: At any point in time, you may store only one datafile record in main memory; at no point in time you can have all the records stored simultaneously in main memory. Our assumption is that the datafile may be too big, so it cannot be fully stored in main memory. However, our assumption is that you can store the primary key and byte offset of all records, simultaneously, on the index array in main memory.

Your project should be called `idf` (short for indexed datafile). Your program should get the datafile name (input and output) and the operations file name (input), from the command line, similarly to assignment#1:
`idf datafile1.dat operations1.txt`

Special Warnings: 1) You may need to use the `clear()` method of `fstream class` various times after positioning in the datafile to clear the end-of-file flag. 2) You should keep a copy of the original datafile you created, since every time you run your program, it gets changed.

Documentation and Style
Your program must be well-documented and written in good style. Part of the marks will be dedicated to documentation and style. Good style includes good documentation (explanation/comments), choice of clear names (for variables, classes, functions, etc) and good program organization and design (like creating methods that deal with each aspect/task within a class).

3 What and how to submit your assignment

Please, read the course's policy on plagiarism and collaboration. You must not include ideas or pieces of code or files from your colleagues or from other programs available in the web or elsewhere. By submitting this assignment you are automatically acknowledging understanding of the policy.
Create a directory/folder named `a2` containing only the following files:
`report.txt` or `report.doc` (report in plain text or word format), `datafile.dat` (your datafile created in part 1, corresponding to the original file (not the modified one)), plus all program files (`.h` and `.cpp`) need to create your project for part 2.
EVERY FILE (except `datafile`) MUST CONTAIN A HEADER WITH Student Name, Student Number, Course and Section. Zip the folder and submit the zipped file as your assignment#2 submission to webCT (only one file is submitted); this should be done in such a way that unzipping will produce folder `a2` with the files inside.