CSI 2131 File Management — Winter 2004
Computer Science — University of Ottawa

Homework Assignment #1 (100 points, weight 10%)

Due: Monday, February 9th at 5:00 p.m. (via webCT)

1 Program: Keysorting (80 marks)

1.1 Problem description

You will write a C++ program that reads from an input file and creates an output file sorted by one of the fields, using a method called "keysorting". The format for each record in the output file is slightly different than the format for the input file, as specified below.

Input file description:
The input file consists of course records; records come in no particular order. Each record has variable length and is terminated by two characters indicating end of line. The fields are mostly of variable length, and separated by the character `|`. The fields in each record are:

- course code (7 characters)
- course title (variable length ≥ 1)
- number of credits (1 character)
- prerequisite information (variable length ≥ 0)
- course description (variable length ≥ 1)

The input file should be read as a binary file.

Output file description:
The output file will have records sorted by course code. Each record has variable length. The records are almost identical to the ones in the output file, except that the records do not contain the two characters (end of line) at the end, and the first two bytes of a record consist of a length indicator containing the number of bytes in the record (not counting the length indicator). The output file should be written as a binary file.

Keysorting method:
It is very important that you follow the keysorting method described here, rather than using another method of your choice.
Keysorting is a method for sorting a file by a specified key. It is used in situations where the main memory is insufficient to hold all the records of a file, but it is large enough to hold all the keys for the file. We will work under this assumption.

Keysorting consists of the following steps:

1. Read the input file, one record at a time, storing in an array only the key (course code) and the record's byte offset. (Note: your array should not hold the full record contents, but just the key)
2. Sort the array in nondecreasing order of key.
3. Using the information in the array, write the records of the output file (sequentially) in sorted order by key. Note that after the array is sorted, you will have to go through the array positions in order. For each position of the array you need to use the byte offset in order to position and read the record again from the input file, and immediately write the record (sequentially) into the output file with the new format.

Note that in the first step, once you have the byte offset for a record and for the previous record, you can calculate the record length, which will be needed when writing the output records in step 3. In C++, you can easily obtain the current get pointer position in a file using the method `tellg`.

1.2 Implementation details and standards for your program

Command line arguments:
The input file name for your programs will be given on the command line. This means that after your program is compiled and an executable file is created, you are going to run it on the DOS prompt and specify some arguments which will represent file names to be used in your program. Details on how to access command line arguments inside your program will be explained in a separate page in the web. In short, the main mechanism to do that involves the use of the following arguments in the main program:
`int main (int argc, char* argv[]) ...`
You may look for information on the use of these parameters on the web or on a C++ book.

Creating and running your program:
The standards below (file names and way of running your program) should be followed strictly. You can use your creativity in the way you design your class and program, but the number and name of files and the way they are used by the teaching assistant should follow the standard.
Create a project and call it `keysorting` with the following classes and files:

- keysort (files: `keysort.cpp, keysort.h`)
- main program (file: `main.cpp`)

The executable program will be called `keysorting.exe` by default, since `keysorting` is the project name.

This program (keysorting.exe) will be executed at the DOS prompt where the user can specify command line arguments using the following format:
`keysorting.exe <inputfilename> <outputfilename>`
or alternatively (the user may omit the extension .exe):
`keysorting <inputfilename> <outputfilename>`
where `<inputfilename>` is the physical name of the input file and `<outputfilename>` is the physical name of the output file.

Testing and checking your output files:
We will provide two input files for testing your program: `small.txt` and courses.txt.
We will use a program that checks your output files. It will check among other things that the output file is sorted and it will move from record to record via the initial length indicator. Therefore, you should create the output file in the exact format that is specified in this handout. When the checker program is available, it will be provided to you at the webCT.

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. To document your program, write comments explaining what is done in the following lines and add comments at the end of certain lines explaining what has been done on that line. For each function or method add a full explanation before it, explaining its purpose and what its parameters represent. For each class created, add an explanation before it. Good style includes good documentation, 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).

What and how to submit your assignment
Create a directory/folder named `a1` containing only the following files: `main.cpp`, `keysort.cpp`, `keysort.h` and `written.txt`. The file `written.txt` contains your answers to the written questions given in Section 2. EVERY FILE MUST CONTAIN A HEADER WITH Student Name, Student Number, Course and Section. Zip the folder and submit the zipped file as your assignment#1 submission to webCT (only one file is submitted).

2 Written Part: Secondary Storage Devices (20 marks)

2.1 Disks (10 marks)

Given a Western Digital Caviar AC22100 Disk System with the following characteristics:

Capacity: 2,100 MB

Average Seek Time: 12 msec

Average Rotational Delay: 6msec

Transfer rate: 12 msec/track

Bytes per Sector: 512

Sectors per track: 63

Tracks per cylinder: 16

Cylinders: 4092

Assume that you want to store a file with 350,000 80-byte records.

1. How many cylinders are necessary to hold the file given that a record can span several sectors, tracks or cylinders?
2. How many cylinders are necessary to hold the file if a record is not allowed to span more than one sector ?
3. Assume the situation in question 2, and assume that there are no interferences from other processes and that the file completely fills a cylinder before it continues into the next cylinder and that the required cylinders are dispersed randomly on the disk. How long would it take to access the entire file in sequence?
4. Under the same assumptions as question 3., how long would it take to access the entire file randomly (accessing every record, but in random order)?

2.2 Tapes (10 marks)

Assume that you want to store a file with 350,000 80-byte records on a 2400-foot reels of 1600-bpi tape with 0.5-inch interblock gaps.

1. What are the minimum and maximum blocking factors that make it necessary to use exactly three tapes?
2. What is the effective recording density when a blocking factor of 50 is used?
3. Given that the tape speed is 150 ips. What is the effective transmission rate of this configuration?