CSI 2131 File Management — Winter 2002
Computer Science — University of Ottawa

Homework Assignment #1 (100 points, weight 10%)
Due: Tuesday, February 5, 2:00 p.m., at the box CSI2131A or CSI2131B (MCD, 3rd floor)

---

1 Problem 1: Programming Assignment (80 marks)

You are probably familiar with Web search engines such as Yahoo!, Altavista, and Google. In this assignment, your task is to design your own simple search engine for a non-connected mini version of the Web. In other words, you will be searching a database of html documents. Your program will take as input a list of html documents[1] and a list of keywords. It will determine the total frequency of the keywords in each document, so that it will print a list of the most relevant documents (i.e., the physical name of the file in which this document is saved) along with a measure of relevance (number of hits).
In more detail, your program will have to do the following tasks:

Filtering words from the html document This part of your program will take as input an html file and it will read from it, so that you extract the next valid word. Your program should disregard html tags (see Note 1 below). Your program should be able to parse words, which are usually followed by white spaces, punctuation or special symbols. For example:
`Hello, world!  This is a great <a href="file1.html"> phrase</a>.`
The words are: `"Hello"`, `"world"`, `"This"`, `"is"`, `"a"`, `"great"`, `"phrase"`.
Also, capitalization should not influence search and frequency counts; the following worlds should be considered the same: `Hello`, `HELLO`, `hello`.

Note 1: Html files contain both opening and closing tags and words. All opening html tags appear between the following pairs of brackets: "<" and ">". All closing html tags appear between the same pair of brackets, plus a slash, "/". For example, `"<B>Hello</B> World"` will print `"Hello"` in boldface characters and `"World"` in regular characters. Both `"<B>"` and `"</B>"` should be filtered out, though neither `"Hello"` nor `"World"` should. Other tags are more complicated than the ones above. For example:
`<a HREF="/h/usr1/document1.pdf">First document</a>`
will print `"First document"` and link it to a hyperlink. Once again, all of `"<a HREF="/h/usr1/document1.pdf">"` and `"</a>"` should be discarded but neither `"First"` nor `"document"` should.

Matching keywords and calculating the relevance of each document For each html document and each valid word in them, you should check whether it matches any of the given keywords. The relevance of the html document will be measured as the number of times matching keywords appear in the document (number of hits).

[1]See below for a description of what an html document is.

First, you calculate the number of hits (relevance) of each of the given documents. Then, you sort the documents by number of hits, and print the ones that have had some hits, in decreasing order of hits. (You may use any sorting algorithm implemented by you or available in the compiler's library, such as `qsort()`.)

For example, suppose the given keywords are: `"art"` and `"school"`, the html documents are: `"doc1.htm"`, `"doc2.htm"`, `"doc3.htm"`, `"doc4.htm"` and `"doc5.htm"`, and that `"art"` appears 4 times in `doc1.htm`, twice in `doc4.htm`, and doesn't appear in the other files; and `"school"` appears twice in `doc2.htm`, 3 times in `doc4.htm`, and doesn't appear in the other files.

The output of your program should be:

```
html documents: doc1.htm, doc2.htm, doc3.htm, doc4.htm, doc5.htm
keywords: art, school
results:
   doc4.htm, <hits= 5>
   doc1.htm, <hits= 4>
   doc2.htm, <hits= 2>
```

Note: marks will be given for programming style (good choice of variable and procedure names, good organization into classes and procedures, presence of explanatory comments, program being well written, etc.)

Standards for your program and what to hand in

Marks will be deducted if you don't follow these standards.
Input and output should be standard. Use the partial code provided by us in the course web page: `a1.cpp`. In this code, the input of the html file names and keywords is already done for you and the output (as described above) is partially done for you.

Your program will be compiled and tested with Microsoft Visual C++ compiler. Failure to compile or run under this compiler will be consider as a failure in your code.

Your project should be called `"a1"` and contain only the following files:

- a1.cpp: main program and any other auxiliary procedures
- htmldoc.h, htmldoc.cpp: definition and implementation for a class that manipulates an html document.
  Details on how to implement this class are up to the programmer, but we give some suggestions. Each html file would be associated to an instance (object) of this class. The class may store the html file name, and may contain: a method to read the next valid word in the html document, a method that returns the number of hits given an array of keywords, among other methods.

(standards continue on the next page)

You should hand in, on a labeled large envelope:

- stapled sheets of paper of your handwritten or typed answer for Problem 2;
- a printout of every file in your program `(*.h, *.cpp)`;
- a printout of the output of your program for a standard input (html documents and keywords provided by us) available from the course web page;
- a 3.5-inch DOS/Windows diskette containing only: the project workspace with all the relevant files in the project `(*.h, *.cpp, *.exe)`.

The envelope as well as all pages handed in should contain: student name, student number, course code (CSI2131), section (A or B), instructor's name, lab group, lab TA name.
All programs handed in the diskette should also contain all the information mentioned above (student name, student number, etc); the diskette itself must be labeled with the same information.

2 Problem 2: Written Assignment (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?