CSC B09 Assignment 2, Winter 2018

Due by the end of Friday March 2, 2018; no late assignments without written explanation.

1. Removing HTML tags

It is difficult to read the content of the following text:

```
<font name="Moronicity" size=12><!-- ignore this comment --><i></i>
<div style="my stupid style sheet">hello</div></font><img src="spacer.gif">
<div style="something else totally useless"><img src="spacer.gif"></div>
```

However, if you pass it through my "unhtml" program, you get:

```
hello
```

Write the unhtml program, which removes HTML tags from its input data. You will use a simple algorithm: copy data until you hit a '<', then discard data up to and including a '>', then resume copying, and so on.

There is no need to use getopt() for this program, but you do need to loop through the filename arguments and process them all in the usual way. If there are no filename arguments, you process the standard input.

As always, the argument files are not modified; the revised text (and nothing else!) is simply output to the standard output.

2. Word-oriented version of *fold*

Write a standard unix tool to break long lines in a text file, except that your tool will break long lines at a space if possible rather than in the middle of a word.

This tool takes file names on the command line or it reads the standard input, in the standard unix tool manner, and outputs on the standard output.

The default width is 80. There is a command-line option which specifies the width (−w plus a numeric argument), and another command-line option −p which says to be willing to break the line after any punctuation mark even in the absence of spaces.

Examples:
For the input text

```
I like the unix programming environment,really I do
```

the command "foldw −w17" would produce the four lines

```
I like the unix
programming
environment,reall
y I do
```

and the command "foldw −w17 −p" would produce the four lines

```
I like the unix
programming
environment,
really I do
```

The `-w` and `-p` flags must work in either order and in any standard combination. Use *getopt*() to process the command-line options correctly.

(over)

3. Directory traversal: largest file

Write a program in C named “maxfile” which takes zero or more command-line arguments which specify directories. Its output is a single line containing the path name of the largest file in any of those directories. If there are no command-line arguments, it searches the current directory (and below).

The path name must be output in terms of the argument to maxfile. For example, if you run “maxfile foo”, the output would look more like “foo/bar/bigfile” than like “/u/ajr/foo/bar/bigfile”. Just concatenate the additional path name components to whatever string the user specifies.

You may impose a maximum path length of 1000 chars, but you must not exceed array bounds even if this path length would be exceeded (you can, however, abort with an error message in such cases, rather than having to deal with longer path names).

You may not use ftw(), fts(), nftw(), or similar library functions (the point of this assignment question is to *implement* the filesystem-tree traversal).

Other notes

Your programs must be in standard C. They must compile on the UTSC linux machines with “gcc −Wall” with no errors or warning messages, and may not use linux-specific or GNU-specific features.

Pay attention to process exit statuses. Your programs must return exit status zero or one as appropriate.

Call your assignment files unhtml.c, foldw.c, and maxfile.c. Your files must have the correct names so as to be processed correctly by the grading software, and auxiliary files are not permitted. Submit with the command

```
submit -c cscb09w18 -a a2 unhtml.c foldw.c maxfile.c
```

and the other ‘submit’ commands are as before.

Please see the assignment Q&A web page at

```
https://mathlab.utsc.utoronto.ca/courses/cscb09w18/a2/qna.html
```

for other reminders, and answers to common questions.

Remember:

This assignment is due at the end of Friday, March 2, by midnight. Late assignments are not ordinarily accepted, and *always* require a written explanation. If you are not finished your assignment by the submission deadline, you should just submit what you have, for partial marks.

Despite the above, I’d like to be clear that if there *is* a legitimate reason for lateness, please do submit your assignment late and send me that written explanation.

And above all: Do not commit an academic offence. Your work must be your own. Do not look at other students’ assignments, and do not show your assignment (complete or partial) to other students. Collaborate with other students only on material which is not being submitted for course credit.