CS 218
Homework, Asst. #11 – Part A

Purpose: Become more familiar with operating system interaction, file input/output operations, and file I/O buffering.
Due: Monday (6/27)
Points: 225 (grading will include functionality, documentation, and coding style)

Description:

Write a program that will open an HTML[1] formatted text file, read the text information, remove all HTML formatting commands, and write the resulting text to another file.

The program should use command line qualifiers to determine the input file and output file names. For example, assuming the executable file is named "page.htm":

```
ed-vm% ./remover -i page.htm -o page.txt
```

Which would open and read the text file "asm.htm", remove all HTML formatting commands, and write the resulting text to a file named "page.txt".

Note, for assignment #11, an HTML formatting command is considered to be all information contained within the "<" and ">" symbols . The "<" and ">" characters, and all information in-between, should ***not*** be written to the output file.

To ensure overall efficiency, the program **must** perform buffered input with a buffer size of BUFF_SIZE (initially set to 500,000). *Note*, the parameter should be used, as the value will be changed in part B. The file buffering should be fully implemented within the ***getBlock()*** function.

The program is expected to perform basic error checking and issue appropriate error messages. For example, if the command line arguments are missing/incorrect or the file names are illegal/invalid/nonexistent, an appropriate error message (pre-defined) should be displayed. In order to complete this program you will need to use the provided main which calls the following routines.

- A boolean function, ***getFiles()***, to read the command line qualifiers and verify file names (by attempting to open the files). If successful, returns the file descriptors and TRUE, otherwise displays an error message and returns FALSE.
- A value reurning function, ***getBlock()***, to return a block of characters from the input file. The routine must perform buffered input and read a buffer full of characters and return one block at a time. As such, the routine will fill and re-fill the block buffer as required. The routine will return the actual number of characters in the block buffer.
- A value reurtning function, ***editBlock()***, to accept a block of text, remove all HTML formatting, and return the edited block and the count of characters in the edited block.

1 For more information, refer to: https://en.wikipedia.org/wiki/HTML

- A void function, ***putBlock()***, to write the block to the output file. The routine is not required to perform buffered output.

Note, your functions must be placed into a *separate file* and linked with the provided main.

Updated Compile, Assemble, and Linking Instructions

When compiling, assembling, and linking the files for assignment #11, use the provided script file. *Note*, **only** the functions file will be submitted. The submitted functions file will be assembled and linked with the provided main. ***As such, you must not change the provided main!***

Submission:

When complete, submit:

- *A copy of the* ***source file*** *via the class web page (assignment submission link) by 11:55 PM.* ***Assignments received after the due date/time will not be accepted.***

Testing

A script file to execute the program on a series of predefined inputs will be provided. *Note*, please follow the I/O examples. The test utility should be downloaded into an empty directory and the program executable placed in that directory. The test script, named **`a10tst`**, can be executed as follows:

```
./a11tst remover
```

The test script compares the program output to predefined expected output (based on the example I/O). As such, you should not change the provided error/status message strings.

Example Executions:

The following execution will read file "tst.htm" and create a file named "tst.txt".

```
ed-vm% ast11 tst.htm tst.txt
```

If the input file, ***tst.htm***, contains the following:

```
<title example></title>Hello World!
<br><p>Hello <b>World!</b></p>
<br><p>Bye</p>
<br><p>end</p>
<br>
```

The output file, ***tst.txt***, should contain the following lines:

```
Hello World!
Hello World!
Bye
end
```