CS xyz3-001 Foundations of Programming and Data Structures

Instructor Dr. Turgay Korkmaz

Homework 09
Due date: check BB
!!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

(Command-line Arguments, Files, Abstract Data Type - Library)

You are asked to write a new application program to check if the HTML tags in a given file are nested correctly or not. This problem is similar to the example of proper matching of `{ ( [ ] ) }`, but now you will deal with HTML tags and read them from a file. So you need to use stack ADT as we discussed in class and make sure you get (stack.h and stack.c) from the slides and make them work with a simple application first!

Here is some Background information about HTML files:
HTML files consist of regular text and **tags** enclosed in angle brackets, the **<** and **>** symbols. Tags are used to identify the structure of a document.

Most tags come in pairs: a **beginning** tag and a **closing** tag. For example, the tags `<title>` and `</title>` are the beginning and closing tags. There are several such tags including `<b> </b> <i> </i> <h1></h1>` etc.

Tags may have several attributes and such attributes will be in the beginning tag. For example `<a href="ff.html"> link </a>`.

HTML allows two-sided tags to be nested without overlapping, as in the example of proper matching of `{ ( [ ] ) }`. In this assignment, they you will use the same idea to check if the HTML tags are properly nested or not using stackADT.

Please note that some tags are single-sided (e.g., `<img src="a.jpg" />, <br />, <hr />`) and they appear alone. They will not affect the nesting but you still need to process them as they might be in the file.
You can simply ignore the tags that start with `<` and end with `/>`

Also the file may contain HTML comments such as

```
<!-- comments contain <p> <\p> -->
```

Your program should ignore everything between `<!--` and `-->` and assume that there will be no nested HTML comments.

Your program must accept HTML input file name as a command line argument and then process it for proper nesting of HTML tags. So we will run your program as follows

```
fox03> driver sample1.html
```

You can stop the program when you detect the first tag that violates the proper nesting structure and print NO, which violates nesting, and the tags that are still in the stack. Otherwise, your program will print YES, all the tags are nested correctly.

For example, if an HTML file `sample1.html` contains

```
<title><b> THIS FILE </b> USES CORRECTLY NESTED TAGS
</title>
<h1><i> First <b class="c1"> header </b>
text <img src="pic.jpg" /> </i></h1>
<!-- comments <p> <b> <\p> all must be ignored
    more comments still ignore  -->
<p id=par1"> Some other text </p>
```

Then your program should print

```
YES, all the tags are nested correctly.
```

But if the HTML file `sample2.html` contains

```
<div> <title> <b> THIS FILE </title> IS </b> NOT NESTED
CORRECTLY.
<p> <b>  some text is not nested correctly </b>
```

Then your program should print

```
NO, the tag <b> and </title> violates the proper nesting!
The tags that are currently in the stack are:
<title>
<div>
```

To implement your application you can use the stackADT that we implemented in class, which is available at the class web page. For this, first get `booklib.zip` and `08-Abstract-Data-Types.zip` from the class web page by clicking the link "programs from the textbook" under **online materials** in the class web page. Then click 00-zip-files. Get README.txt and read it. Accordingly, get the other files as described in README.txt]

The implementation in 08-Abstract... has the stack library implementation and a driver program (rpncal.c) for RPN calculator. During the recitation you will modify rpncal.c. But for this homework, you need to write a new application program (say htmlchecker.c), which will use the stack lib as rpncal.c does. So you will use the existing implementation of stack library! But , in your new application, since you will push/pop html tags (strings), you need to make sure stack.h has `typedef void *stackElementT;` // or
`typedef char *stackElementT;`

As always, make sure you release (free) the dynamically allocated memories if you allocate any memory in your programs. So, before submitting your program, run it with `valgrind` to

see if there is any memory leakage… Also if you need to debug your program, compile your programs with –g option and then run it with `gdb` and/or `ddd`.

Here are some explanations for the questions in previous years:

You can assume that the max length of a tag will be less than 250 characters..

You can assume that the whole tag will be in one line but the beginning and closing tags could be in different lines like
<title>
....
</title>

so we will not have something like <abc .....

> .....

But Comments <!-- ..... --> could be in one line or in multiple lines like

<!-- .....

..... -->

you have to ignore everything between <!-- and --> and there will be no other comment in a comment (no nested comments) !

What to do and return: !!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

1. Create a directory `abc123-hw09`, using your own abc123. Do all your work under that directory. First get stack.h and stack.c work with a simple driver program.

2. To easily compile the stack library and driver program, you must have a Makefile and use "make" to compile your code.

3. Follow the problem solving methodology, and solve the problem(s). Then convert your solution(s) to a C program usign stack ADT. You can name your program here as `hw09.c`

```
/*
 * Don't forget to include comments about the
 * problem, yourself and each major step in your
 * program! so that we can understand your
 * solution(s).
 */
```

4. Compile your program using Makefile. Then run it and copy/paste the results in an output file, which you can name as hw09-out.txt. Also make sure you get hw09-valgrind.txt, as described in previous assignments.

5. Zip the whole directory `abc123-hw09 as abc123-hw09.zip`
6. Go to BB Learn (http://learn.utsa.edu/) , login using your abc123
7. Submit your `abc123-hw09.zip` for hw09 under Assignments

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as either a zip or tar archive file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include an output.txt file which demonstrates the final test output run by the student.
4) If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.
