Project 2: Basic HTML Parsing and Crawling

Instructions: This project is divided into two parts:

1. build a simple HTML parser that determines if your HTML tags are balanced,
2. find the number of webpages you can visit from a certain HTML page.

To keep things simple for this project, all pages will be local to your machine.

Part 1: Basic HTML parsing

It is important to know if your HTML tags are balanced. For example:

```
<html>
    <body>
        <b>
            Hello World!
        </b>
    </body>
</html>
```

is balanced since each tag has a begin tag (<tagname>) followed by an end tag (<tagname>) at the same "level depth". For instance, consider the following:

```
<html>
    <body>
        <b>
            Hello World!
        </body>
    </b>
</html>
```

The above is not balanced because there is a < /body> ending a <b> tag. Given the following basic HTML definition your job is to determine if a given piece of HTML is balanced or not.

HTML	<html>BODY< /html>
BODY	<body>TEXT< /body>
TEXT	STRING — STRING TEXT — TAG — TAG TEXT
STRING	possibly empty string of printable characters other than '< and '>
TAG	BOLD — ITALICS — LINK
BOLD	<b>TEXT< /b>
ITALICS	<i>TEXT< /i>
LINK	<a href=URL>TEXT< /a>
URL	STRING

Part 2: Basic Web Crawler

Now that we have a basic HTML parser, write a web crawler to determine the number of unique pages that can be visited from a certain page. See example below:

Example Output:

```
./html-test pages/pokemon.html pages/theend.html pages/notbalanced.html pages/index.html
Parsing: 'pages/pokemon.html'
Parsing: 'pages/theend.html'
Parsing: 'pages/notbalanced.html'
Parsing: 'pages/index.html'
pages/pokemon.html is balanced.
pages/pokemon.html can visit 1 pages.
pages/theend.html is balanced.
pages/theend.html can visit 0 pages.
pages/notbalanced.html is not balanced.
pages/notbalanced.html can visit 0 pages.
pages/index.html is balanced.
pages/index.html can visit 2 pages.
```

pokemon.html, theend.html, and index.html are balanced while notbalanced.html is not. Since notbalanced.html is not balanced, we say it can visit 0 pages since it does not parse. Also, while an example is not given, if the link is not visit-able (IE the page does not exist), then do not count towards a possible link visit. Here is the explanation of the visit amounts:

- theend.html has no links in it, so the visit amount is 0.
- notbalanced.html has no links since it is not balanced.
- pokemon.html can visit one page (pages/theend.html) which is valid and a dead end.
- index.html can visit two pages even through it has one link. The link goes to pages/pokemon.html, which is valid, and that page can visit another page pages/theend.html.

Keep in mind the visit amount is the amount of **unique** pages that can be visited from a certain page.

Parsing:
The fist part of this project requires you to read a file and parse out the input. This may seem daunting at first, but here are a couple of things to help out:

- Tags:
  - begin with a "<" and end with a ">".
  - do **not** have a < or > between the starting < and ending >. Therefore once you see a < parse to a > and that string is your tag.
  - All tags, except the anchor tag (<a>), will have no attributes or whitespace. That means the body tag will always be "<body>", there will be no spaces or other characters. The same goes for the end tags.
  - The anchor tag will **strictly** be in the form "<a href=filename>" In other words only one space will be there between the a and href.
- If you encounter a tag that is not valid then the page is not balanced.
- I will ensure all tags are lower case.

- To ignore whitespace, parse everything one character at a time ignoring space, tab, newline, and carriage return. (" \t\n\r")

Hints:
So far we have used arrays, linked lists, pointer, stacks, and queues. Chances are you will be using more that one data structure to solve this problem, but you will only need the ones listed.

Also, if you are familiar with the STL, you may use vector, stack, and queue from the STL library, but nothing else.

Warning:
Data Structure projects are *significantly* more demanding than labs. It is imperative you start thinking about the data structure now! If you hit a road block (and you should/will), please come talk to me. When you talk to me be prepared to show me what you have thought of so far. If you come in with an idea I will guide your idea to a good solution. If you come in without and idea, I will tell you to keep thinking about it. Keep in mind there are many good, fair, and terrible solutions to Data Structure projects.

Write some test cases:
Part of this project grade will be how well you can write test cases. You will be in charge of all test cases.

How to turn in:
Turn in via GitHub. Ensure the file(s) are in your directory and then:

- $ git add <files>
- $ git commit
- $ git push

Due Date: March 11, 2019 2359

Teamwork: No teamwork, your work must be your own.