Practice Exercise #32: Simple Parser

http://www.comp.nus.edu.sg/~cs1020/4_misc/practice.html

(This exercise is Sit-in Lab #3 Set B of AY2012/2013 Semester 2)

Objective: Using LinkedList or Stack class

Background

A language designer has designed a new markup language for the web. In order to render a source file written in the markup language as a web page, he requires a program called the parser to check that the source file is syntactically correct. You have been assigned to write this parser.

Note: You are required to make use of the LinkedList ADT or any of its variants, or the Java API Stack, in solving this problem.

Task Statement

The markup language consists of a few simple **tags**. Any text string with '<' or '</' immediately to its left and '>' immediately to its right is taken to be a tag, whether it is a valid tag or not. Everything that is not a tag is simply known as **text**. The valid tags can be categorized as

- Paired Tags – Tags that come in pairs, one being the opening tag and the other being the closing tag. We call the space between an opening tag and its corresponding closing tag an **environment**. Text and any tag can be embedded in the environment of a paired tag. Since paired tags can be embedded within other paired tags, there can be possibly many levels of embedding.
- Non-Paired Tags – Tags that do not come in pairs.

The valid tags designed by the language designer are as follows:

Paired Tags	Opening Tag	Closing Tag
Section	<S>	</S>
Paragraph	<P>	</P>
Bold Font	<B>	</B>
Italic font	<I>	</I>

Non-Paired Tags	Tag
Line Break	<LB>
Page Break	<PB>

There is a special tag <TEXT> which is used to represent actual text in the source file, and this, instead of the text, will be passed to your parser.

The job of the parser that you write is then to check that a given source file is syntactically correct. A syntactically correct source file must adhere to the following rules:

1. An empty source file is syntactically correct.
2. An opening tag must be closed. That is, its corresponding closing tag must exist somewhere in the source file.
3. There should not be any invalid tags in the source file.
4. If an opening tag is embedded in the environment of another opening tag, the inner opening tag must be closed before closing the outer opening tag.

Input

The target file that your parser will work on has already been tokenized into tags and will be passed as input to your parser. As mentioned, text is not passed directly but rather is represented by the tag <TEXT> in the input. The input then consists of some lines, where each line is a tag (whether valid or not). Note that the number of lines is not indicated in the input. An example of an input which is syntactically correct is shown below.

```
<S>
<P>
<TEXT>
</P>
<LB>
<P>
<TEXT>
<TEXT>
</P>
<PB>
</S>
```

An example of an input which is syntactically incorrect is as follows.

```
<S>
<KB>  ← no such tag!
<P>
<TEXT>
</P>
</S>
```

Another example of an input which is syntactically incorrect is as follows.

```
<S>
<P>  ← missing its closing tag!
<TEXT>
<P>
```

```
<TEXT>
</P>
</S>
<S>
<P>  ← must be closed before closing <S>
<TEXT>
</S>
</P>
```

Output

Simply print out **No Error** if the input is syntactically correct, or **Error!** if it is not.

Skeleton Program

Your program is to be named **Parser.java**. A skeleton program is provided, but you can change it any way or add new class(es) as you deem fit when you write your program. In the skeleton program, the ListNode <E> class is provided as well as a very rudimentary LinkedList <E> class (if you want to use this class you will have to complete it). If you choose to use the **Stack** API, you may remove the code related to **ListNode** and **LinkedList** from your program.