Complexity Analysis -- Assignment 3

Data Structures (Section A & B)

Fall 2018

Due Date: 12:00 pm, 12th October 2018

Submission Location: Upload the zip file containing your solution on the Google classroom. The name of the ZIP file should be your Roll Number.

QUESTION: Design a C++ program that use a stack for matching tags and quotes in XML eXtensive Markup Language.

Your program will get an XML code in an input file and it should figure out if tags and quotes are properly matched or not using stack. In case the tags are not properly matched then your program should report following

- **i) the first error,**
- **ii) print the mismatched tag and**
- **iii) inform the line number where the starting tag occurred.**

What is XML? XML is a markup language somewhat like HTML. It was designed to store and transport data. XML is just information wrapped in user defined tags which is both human- and machine-readable. The XML language has no predefined tags like html. The tags are "invented" by the author of the XML document. For details see https://www.w3schools.com/xml/xml_whatis.asp

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <message>Don't forget me this weekend!</message>
</note>
```

In above example
<?xml version="1.0" encoding="UTF-8"**?>**
This is XML prolog (header). It starts with <? and ends with ?>. The header should come in the start of the document.

<note>, <from>, <heading> and <message> are user defined tags and each must have a corresponding ending tag.

Your program should handle the following features of XML:

1. **xml prolog (xml header)**
2. **xml tags (xml elements). The xml tags are case sensitive.**
3. **xml attribute**
4. **xml comments, start with <!-- and ends with -->**

Consider another example

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
 <book category="cooking">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
 </book>
 <book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
</bookstore>
```

In the example above:
<title>, <author>, <year>, and <price> have **text content** because they contain text (like 2005).
<bookstore> and <book> have **element contents**, because they contain elements.
<book> has an **attribute** (category="children").

Note that Attribute values must always be quoted. Either single or double quotes can be used.

Your program should keep track that attributes have opening and closing quotes.

Your code will have template-based Node class and template-based Stack class. Implement stack using singly linked list.

```
template<class T>
class Node {
public:
       Node()
       T data;
       Node<T> * next;

};
```

```
template<class T>
class Stack {
public:
       Stack() ;
       ~Stack();
       bool IsEmpty();
       bool push(const T & val);
       bool pop(T & val);
       T top();
       void print();
private:
       Node<T> * top;

};
```

Create XMLData class
Think about its attributes carefully

```
Void main(){

   Stack<XMLData> S1;
}
```

CODE DESIGN GUIDLEINES

- Do template-based programming
- Code should be properly intendent and commented (2 marks for this)
- Make sure there are no memory leaks or dangling pointers
- Don't cheat or take too much unnecessary help from your friends