# Checking XML consistency:

**Text

Description automatically generated1) xml error detection function:**

* Implementation: this function detects some errors related to xml consistency such as any missed closed tag and any wrong named closed tag and fills the queues with result of detection.
* Time and space complexity:
  + Time: O(n)
  + Space: O(n) (using stack)

**Text

Description automatically generated2) xml error correction function:**

* Implementation: this function corrects the errors in linked list based on queues filled by the error detection function.

* Time and space complexity:
  + Time: O(n)
  + Space: O(1)

**3) xml correct function:**

Text

Description automatically generated

* Implementation: this function returns true if the xml doesn`t include any of the mentioned errors.
* Time and space complexity:
  + Time: O(n)
  + Space: O(n)

**Text

Description automatically generated4) xml error positions:**

* Implementation: this function is called with linked list and queue parameters and its functionality to filled them with the positions of deleted and inserted closed tags and the closed tags to be inserted.

* Time and space complexity:
  + Time: O(n)
  + Space: O(n)

**5) xml automatically correct function:**

Text

Description automatically generated

* Implementation: this function is called with linked list (which includes the parsed string) as parameter and returned it after correcting.

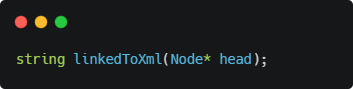
* Time and space complexity:
  + Time: O(n)
  + Space: O(n)

# 6) The conversion functions

## Background

There are two conversion functions used in the checking of the xml one to convert the xml to linked list and the other to convert the linked list back to xml string after correction.

## Parameters and Return

* **xmlToLinked**
* The parameters: xml string and pointer to a linked list
* The return type: void
* **linkedToXml**
* The parameters: the head of a linked list
* The return type: xml string

## Implementation

* **xmlToLinked**: convert the xml string to linked list have nodes contain open tag, closed tag, and text included inside two tags with the same order in the xml string.
* **linkedToXml:** append each node data in xml string and return it.

## Time and Space complexity

* **xmlToLinked**: O(n), n is the number of characters in the xml output string
* **linkedToXml**
* Time: O(m), m is number of nodes in the linked list (tags and texts)
* Space: O(n), n is the number of characters in the xml output string

## The Data Structure used

* **String:** which uses vector contains a **dynamic array** object.
* **Linked list**
* **Stack:** implemented with linked list (list) and has additional function
* **Search(string closeTag):** this function search in stack for the open tag that match the closed input tag. It has time complexity of O(m), m is elements in the stack and space of O(1).