**Introduction to SGML:**

**Elements**

The technical term used in the SGML standard for a textual unit, viewed as a structural component, is element.

Within a marked up text (a document instance), each element must be explicitly marked or tagged in some way. The standard provides for a variety of different ways of doing this, the most commonly used being to insert a tag at the beginning of the element (a start-tag) and another at its end (an end-tag). The start- and end-tag pair is used to bracket off the element occurrences within the running text, in rather the same way as different types of parentheses or quotation marks are used in conventional punctuation. For example, a quotation element in a text might be tagged as follows:

> ... Rosalind's remarks <quote>This is the silliest stuff that ere I heard of!</quote> clearly indicate ...

As this example shows, a start-tag takes the form `<name>`, where the opening angle bracket indicates the start of the start-tag, “name” is the generic identifier of the element which is being delimited, and the closing angle bracket indicates the end of a tag. An end-tag takes an identical form, except that the opening angle bracket is followed by a solidus (slash) character, so that the corresponding end-tag would be `</name>`.

The actual characters used for the delimiting characters (the angle brackets, exclamation mark and solidus) may be redefined, but it is conventional to use the characters used in this description.
Content Models: An Example

An element may be empty, that is, it may have no content at all, or it may contain simple text. More usually, however, elements of one type will be embedded (contained entirely) within elements of a different type.

To illustrate this, we will consider a very simple structural model. Let us assume that we wish to identify within an anthology only poems, their titles, and the stanzas and lines of which they are composed. In SGML terms, our document type is the anthology, and it consists of a series of poems. Each poem has embedded within it one element, a title, and several occurrences of another, a verse, each verse having embedded within it a number of line elements. Fully marked up, a text conforming to this model might appear as follows:

```xml
<anthology>
    <poem><title>The SICK ROSE</title>
    <verse>
        <line>O Rose thou art sick.</line>
        <line>The invisible worm,</line>
        <line>That flies in the night</line>
        <line>In the howling storm:</line>
    </verse>
    <verse>
        <line>Has found out thy bed</line>
        <line>Of crimson joy:</line>
        <line>And his dark secret love</line>
    </verse>
</poem>
</anthology>
```
Considering our greatly over-simplified model of a poem, we could state the following rules:

- An anthology contains a number of poems and nothing else.
- A poem always has a single title element which precedes the first verse and contains no other elements.
- Apart from the title, a poem consists only of verses.
- Verses consist only of lines and every line is contained by a verse.
- Nothing can follow a verse except another verse or the end of a poem.
- Nothing can follow a line except another line or the start of a new verse.

From these rules, it may be inferred that we do not need to mark the ends of verses or lines explicitly. From rule 2 it follows that we do not need to mark the end of the title---it is implied by the start of the first verse. Similarly, from rules 3 and 1 it follows that we need not mark the end of the poem: since poems cannot occur within poems but must occur within anthologies, the end of a poem is implied by the start of the next poem, or by the end of the anthology. Applying these simplifications, we could mark up the same poem as follows:
Before considering these rules further, you may wish to consider how text marked up in the form above could be processed by a computer for very many different purposes.

- A simple indexing program could extract only the relevant text elements in order to make a list of titles, or of words used in the poem text.
- A simple formatting program could insert blank lines between stanzas, perhaps indenting the first line of each, or inserting a stanza number.
- Different parts of each poem could be typeset in different ways.
An Example DTD

A DTD is expressed in SGML as a set of declarative statements, using a simple syntax defined in the standard. For our simple model of a poem, the following declarations would be appropriate:

```xml
<!ELEMENT anthology      - - (poem+)>
<!ELEMENT poem           - - (title?, verse+)>
<!ELEMENT title          - O (#PCDATA) >
<!ELEMENT verse         - O (line+) >
<!ELEMENT line           O O (#PCDATA) >
```

These five lines are examples of formal SGML element declarations. A declaration, like an element, is delimited by angle brackets; the first character following the opening bracket must be an exclamation mark, followed immediately by one of a small set of SGML-defined keywords, specifying the kind of object being declared.

The five declarations above are all of the same type: each begins with an ELEMENT keyword, indicating that it declares an element, in the technical sense defined above.

Each consists of three parts:
- a name or group of names
- two characters specifying minimization rules
- a content model.

Components of the declaration are separated by white space, that is one or more blanks, tabs or newlines.
The first part of each declaration above gives the **generic identifier** of the element which is being declared, for example poem, title, etc.

**Minimization Rules**
These rules determine whether or not start- and end-tags must be present in every occurrence of the element concerned. They take the form of a pair of characters, separated by white space, the first of which relates to the start-tag, and the second to the end-tag. In either case, either a hyphen or a letter O (for "omissible" or "optional") must be given; the hyphen indicating that the tag must be present, and the letter O that it may be omitted. Thus, in this example, every element except `<line>` must have a start-tag. Only the `<poem>` and `<anthology>` elements must have end-tags as well.

**Content Model**
The third part of each declaration, enclosed in parentheses, is called the content model of the element, because it specifies what element occurrences may legitimately contain. Contents are specified either in terms of other elements or using special reserved words. There are several such reserved words, of which by far the most commonly encountered is `#PCDATA`, as in this example. This is an abbreviation for **parsed character data**, and it means that the element being defined may contain any valid character data.

If an SGML declaration is thought of as a structure like a family tree, with a single ancestor at the top (in our case, this would be `<anthology>`), then almost always, following the branches of the tree downwards (for example, from `<anthology>` to `<poem>` to `<verse>` to `<line>` and `<title>`) will lead eventually to `#PCDATA`. In our example, `<title>` and `<line>` are so defined. Since their content models say `#PCDATA` only and
name no embedded elements, they may not contain any embedded elements.

**Occurrence Indicators**
Indicates how many times the element named in its content model may occur. There are three occurrence indicators in the SGML syntax, conventionally represented by the plus sign, the question mark, and the asterisk or star.

- The plus sign means that there may be one or more occurrences of the element concerned.
- The question mark means that there may be at most one and possibly no occurrence.
- The star means that the element concerned may either be absent or appear one or more times.

Thus, if the content model for `<verse>` were `(LINE*)`, stanzas with no lines would be possible as well as those with more than one line. If it were `(LINE?)`, again empty stanzas would be countenanced, but no stanza could have more than a single line. The declaration for `<poem>` in the example above thus states that a `<poem>` cannot have more than one title, but may have none, and that it must have at least one `<verse>` and may have several.

**Group Connectors**
The content model `(TITLE?, VERSE+)` contains more than one component, and thus needs additionally to specify the order in which these elements ( `<title>` and `<verse>` ) may appear. This ordering is determined by the group connector (the comma) used between its components.

There are three possible group connectors:
• The comma means that the components it connects must both appear in the order specified by the content model.
• The ampersand indicates that the components it connects must both appear but may appear in any order.
• The vertical bar indicates that only one of the components it connects may appear.

If the comma in this example were replaced by an ampersand, a title could appear either before the stanzas of a <poem> or at the end (but not between stanzas). If it were replaced by a vertical bar, then a <poem> would consist of either a title or just stanzas---but not both!
Attributes

In the SGML context, the word attribute, like some other words, has a specific technical sense. It is used to describe information which is in some sense descriptive of a specific element occurrence but not regarded as part of its content.

For example
<poem id=P1 status="draft"> ... </poem>

The <poem> element has been defined as having two attributes: id and status. For the instance of a <poem> in this example, the id attribute has the value P1 and the status attribute has the value draft.

An SGML processor can use the values of the attributes in any way it chooses:

- A formatter might print a poem element which has the status attribute set to draft in a different way from one with the same attribute set to revised.
- Another processor might use the same attribute to determine whether or not poem elements are to be processed at all.
- The id attribute is a slightly special case in that, by convention, it is always used to supply a unique value to identify a particular element occurrence, which can be used for cross reference purposes.

Like elements, attributes are declared in the SGML document type declaration (DTD), using rather similar syntax. As well as specifying its name and the element to which it is to be attached, it is possible to
specify (within limits) what kind of value is acceptable for an attribute and a default value.

The following declarations could be used to define the two attributes we have specified above for the <poem> element:

```
<!ATTLIST poem
    id       ID                              #IMPLIED
    status   (draft | revised | published)   draft        >
```

The declaration begins with the symbol ATTLIST, which introduces an attribute list specification. The first part of this specifies the element (or elements) concerned. In our example, attributes have been declared only for the <poem> element. If several elements share the same attributes, they may all be defined in a single declaration; just as with element declarations, several names may be given in a parenthesized list. Following this name (or list of names), is a series of rows, one for each attribute being declared, each containing three parts:

- The name of the attribute
- The type of value it takes
- A default value respectively.