CS335: COMPILER DESIGN - Assignment 2

Harsh Bihany (210406)

Problem 1

Consider the following context-free grammar. The terminal symbols are **id** ( ) * , and $. *Function* is the start nonterminal.

$$
\begin{aligned}
\mathit{Function} &\rightarrow \mathit{Type}\ \textbf{id}\ (\mathit{Arguments}) \\
\mathit{Type} &\rightarrow \textbf{id} \\
\mathit{Type} &\rightarrow \mathit{Type}* \\
\mathit{Arguments} &\rightarrow \mathit{ArgList} \\
\mathit{Arguments} &\rightarrow \epsilon \\
\mathit{ArgList} &\rightarrow \mathit{Type}\ \textbf{id}\ ,\mathit{ArgList} \\
\mathit{ArgList} &\rightarrow \mathit{Type}\ \textbf{id}
\end{aligned}
$$

(i) Explain why the grammar is not LL(1)?

A grammar is *left recursive* if it has a non-terminal A such that there exists a derivation $A \xrightarrow{*} A\alpha$ for some string α. In the following grammar we have one such derivation $\mathit{Type} \rightarrow \mathit{Type}*$.

Further a left recursive grammar cannot be LL(1). Hence the above grammar is not LL(1).

(ii) Perform required transformations on the grammar to make it LL(1). You may introduce new nonterminals. Show your transformed grammar.

To make the grammar LL(1), we need two things. Elimination of direct and indirect left recursion and left factoring to eliminate backtracking if needed.

Removing the direct left recursion mentioned earlier. We get

$$
\begin{aligned}
\mathit{Function} &\rightarrow \mathit{Type}\ \textbf{id}\ (\mathit{Arguments}) \\
\mathit{Type} &\rightarrow \textbf{id}\ \mathit{Type}' \\
\mathit{Type}' &\rightarrow *\mathit{Type}' \\
\mathit{Type}' &\rightarrow \epsilon \\
\mathit{Arguments} &\rightarrow \mathit{ArgList} \\
\mathit{Arguments} &\rightarrow \epsilon \\
\mathit{ArgList} &\rightarrow \mathit{Type}\ \textbf{id}\ ,\mathit{ArgList} \\
\mathit{ArgList} &\rightarrow \mathit{Type}\ \textbf{id}
\end{aligned}
$$

Now, after left factoring (required for the nonterminal $\mathit{ArgList}$) we get,

$$Function \rightarrow Type\ \mathbf{id}\ (Arguments) \quad (1)$$
$$Type \rightarrow \mathbf{id}\ Type' \quad (2)$$
$$Type' \rightarrow *Type' \quad (3)$$
$$Type' \rightarrow \epsilon \quad (4)$$
$$Arguments \rightarrow ArgList \quad (5)$$
$$Arguments \rightarrow \epsilon \quad (6)$$
$$ArgList \rightarrow Type\ \mathbf{id}\ More \quad (7)$$
$$More \rightarrow \mathbf{,}\ ArgList \quad (8)$$
$$More \rightarrow \epsilon \quad (9)$$

Remark. The productions are labeled so that they can be referred subsequently.

(iii) Show FIRST and FOLLOW set for the nonterminals in your transformed grammar.

$$\text{FIRST}(\mathbf{T}) = \{\mathbf{T}\} \quad \forall \mathbf{T} \in \{\mathbf{id}, (,), *, \mathbf{,}, \$, \epsilon\}$$
$$\text{FIRST}(Function) = \{\mathbf{id}\}$$
$$\text{FIRST}(Type) = \{\mathbf{id}\}$$
$$\text{FIRST}(Type') = \{*, \epsilon\}$$
$$\text{FIRST}(Arguments) = \{\mathbf{id}, \epsilon\}$$
$$\text{FIRST}(ArgList) = \{\mathbf{id}\}$$
$$\text{FIRST}(More) = \{\mathbf{,}, \epsilon\}$$

$$\text{FOLLOW}(Function) = \{\$\}$$
$$\text{FOLLOW}(Type) = \{\mathbf{id}\}$$
$$\text{FOLLOW}(Type') = \{\mathbf{id}\}$$
$$\text{FOLLOW}(Arguments) = \{)\}$$
$$\text{FOLLOW}(ArgList) = \{)\}$$
$$\text{FOLLOW}(More) = \{)\}$$

(iv) Show the predictive LL(1) parsing table for your transformed grammar. You do not need to show the working of the parser with an example input string.

NON-TERMINALS	**INPUTS**					
	$	(	)	**id**	*	,
Function				(1)		
Type				(2)		
Type'				(4)	(3)	
Arguments			(6)	(5)		
ArgList				(7)		
More			(9)			(8)

Table 1.1: Predictive LL(1) Parsing Table

Problem 2

Consider the following context-free grammar. The terminal symbols are a, b, p, q, r, s, and t. S is the start nonterminal.

$$\begin{aligned} S &\rightarrow LM \mid Lp \mid qLr \mid sr \mid qsp \\ L &\rightarrow aMb \mid s \mid t \\ M &\rightarrow t \end{aligned}$$

(i) Is the given CFG SLR(1)?

After augmentation of the grammar and numbering the productions:

$$S' \rightarrow S \quad (0)$$
$$S \rightarrow LM \quad (1)$$
$$S \rightarrow Lp \quad (2)$$
$$S \rightarrow qLr \quad (3)$$
$$S \rightarrow sr \quad (4)$$
$$S \rightarrow qsp \quad (5)$$
$$L \rightarrow aMb \quad (6)$$
$$L \rightarrow s \quad (7)$$
$$L \rightarrow t \quad (8)$$
$$M \rightarrow t \quad (9)$$

The *FIRST* and *FOLLOW* sets are as follows:

$$\text{FIRST}(\mathbf{T}) = \{\mathbf{T}\} \quad \forall \mathbf{T} \in \{a, b, p, q, r, s, t\}$$
$$\text{FIRST}(S) = \{a, q, s, t\}$$
$$\text{FIRST}(L) = \{a, s, t\}$$
$$\text{FIRST}(M) = \{t\}$$

$$\text{FOLLOW}(S') = \{\$\}$$
$$\text{FOLLOW}(S) = \{\$\}$$
$$\text{FOLLOW}(L) = \{p, r, t\}$$
$$\text{FOLLOW}(M) = \{\$, b\}$$

Now onto the $LR(0)$ canonical collection

$I_0 = \text{Closure}(\{[S' \to \cdot S]\})$
$= \{[S' \to \cdot S],$
$[S \to \cdot LM],$
$[S \to \cdot Lp],$
$[S \to \cdot qLr],$
$[S \to \cdot sr],$
$[S \to \cdot qsp],$
$[L \to \cdot aMb],$
$[L \to \cdot s],$
$[L \to \cdot t],$
$\}$
$I_1 = \text{Goto}(I_0, S)$
$= \{[S' \to S\cdot],$
$\}$
$I_2 = \text{Goto}(I_0, L)$
$= \{[S \to L \cdot M],$
$[S \to L \cdot p],$
$[M \to \cdot t],$
$\}$
$I_3 = \text{Goto}(I_0, q)$
$= \{[S \to q \cdot Lr],$
$[S \to q \cdot sp],$
$[L \to \cdot aMb],$
$[L \to \cdot s],$
$[L \to \cdot t],$
$\}$
$I_4 = \text{Goto}(I_0, s)$
$= \{[S \to s \cdot r],$
$[L \to s\cdot],$
$\}$
$I_5 = \text{Goto}(I_0, a)$
$= \{[L \to a \cdot Mb],$
$[M \to \cdot t],$
$\}$
$I_6 = \text{Goto}(I_3, t)$
$I_9 = \text{Goto}(I_5, t)$

$I_6 = \text{Goto}(I_0, t)$
$= \{[L \to t\cdot],$
$\}$
$I_7 = \text{Goto}(I_2, M)$
$= \{[S \to LM\cdot],$
$\}$
$I_8 = \text{Goto}(I_2, p)$
$= \{[S \to Lp\cdot],$
$\}$
$I_9 = \text{Goto}(I_2, t)$
$= \{[M \to t\cdot],$
$\}$
$I_{10} = \text{Goto}(I_3, L)$
$= \{[S \to qL \cdot r],$
$\}$
$I_{11} = \text{Goto}(I_3, s)$
$= \{[S \to qs \cdot p],$
$= \{[L \to s\cdot],$
$\}$
$I_{12} = \text{Goto}(I_4, r)$
$= \{[S \to sr\cdot],$
$\}$
$I_{13} = \text{Goto}(I_5, M)$
$= \{[L \to aM \cdot b],$
$\}$
$I_{14} = \text{Goto}(I_{10}, r)$
$= \{[S \to qLr\cdot],$
$\}$
$I_{15} = \text{Goto}(I_{11}, p)$
$= \{[S \to qsp\cdot],$
$\}$
$I_{16} = \text{Goto}(I_{13}, b)$
$= \{[L \to aMb\cdot],$
$\}$
$I_5 = \text{Goto}(I_3, a)$

Figure 2.1: LR(0) Canonical Collection

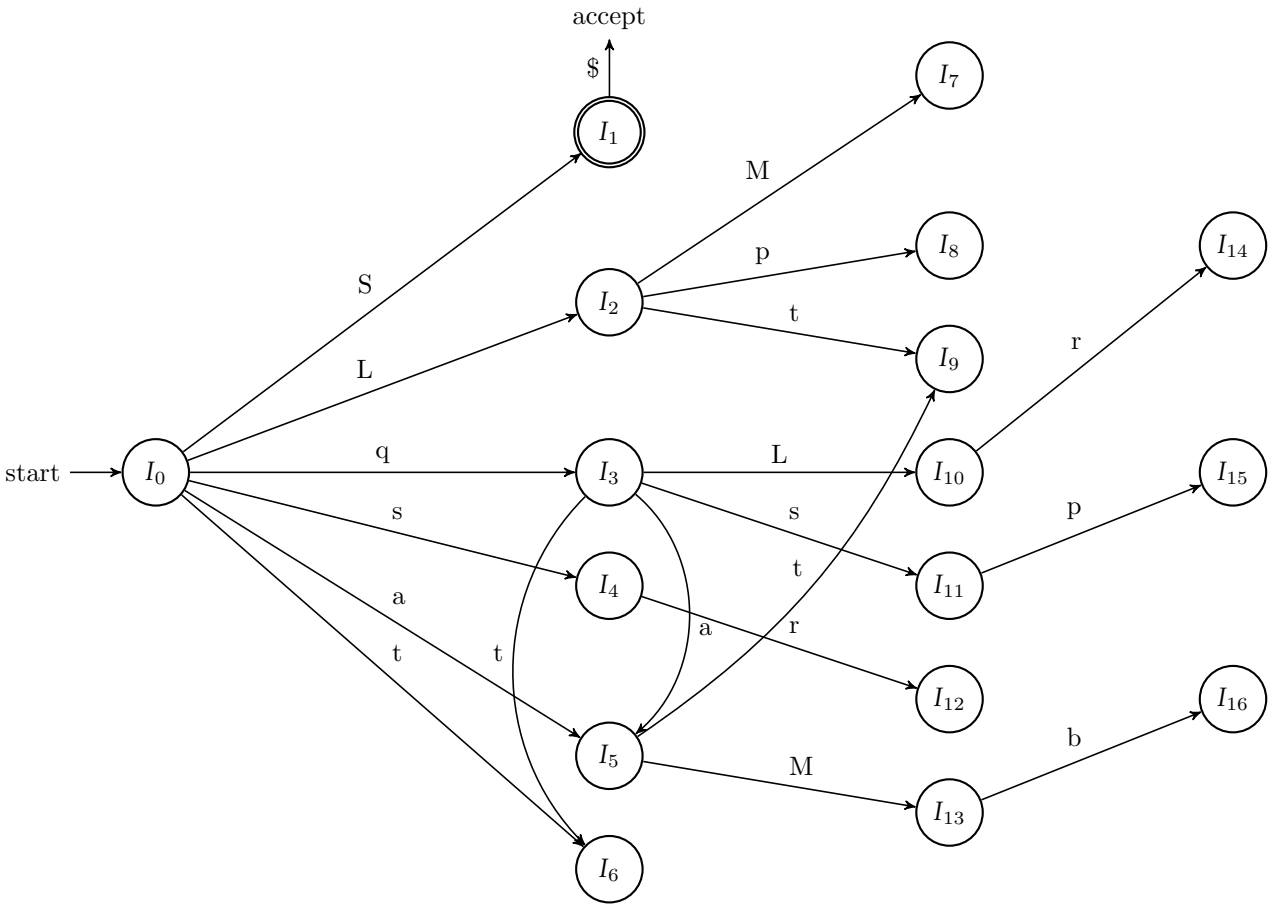

Figure 2.2: LR(0) automaton

STATE	**ACTION**								**GOTO**		
	a	*b*	*p*	*q*	*r*	*s*	*t*	$	S	L	M
0	s5			s3		s4	s6		1	2	
1								**acc**			
2			s8				s9				7
3	s5					s11	s6			10	
4			r7		s12 r7		r7				
5							s9				13
6			r8		r8		r8				
7								r1			
8								r2			
9		r9						r9			
10					s14						
11			s15 r7		r7		r7				
12								r4			
13		s16									
14								r3			
15								r5			
16			r6		r6		r6				

Table 2.1: SLR(1) Parsing table

There are 2 shift/reduce conflicts. Hence the grammar is not SLR(1).

(ii) Is the given CFG LALR(1)?

The *FIRST* and *FOLLOW* sets remain the same.
Now onto the LR(1) canonical set.

$$
\begin{aligned}
I_0 &= \text{Closure}(\{[S' \to \cdot S, \$]\}) \\
&= \{[S' \to \cdot S, \$], \\
&\quad [S \to \cdot LM, \$], \\
&\quad [S \to \cdot Lp, \$], \\
&\quad [S \to \cdot qLr, \$], \\
&\quad [S \to \cdot sr, \$], \\
&\quad [S \to \cdot qsp, \$], \\
&\quad [L \to \cdot aMb, t/p], \\
&\quad [L \to \cdot s, t/p], \\
&\quad [L \to \cdot t, t/p], \\
&\quad \} \\
I_1 &= \text{Goto}(I_0, S) \\
&= \{[S' \to S\cdot, \$], \\
&\quad \} \\
I_2 &= \text{Goto}(I_0, L) \\
&= \{[S \to L \cdot M, \$], \\
&\quad [S \to L \cdot p, \$], \\
&\quad [M \to \cdot t, \$], \\
&\quad \} \\
I_3 &= \text{Goto}(I_0, q) \\
&= \{[S \to q \cdot Lr, \$], \\
&\quad [S \to q \cdot sp, \$], \\
&\quad [L \to \cdot aMb, r], \\
&\quad [L \to \cdot s, r], \\
&\quad [L \to \cdot t, r], \\
&\quad \} \\
I_4 &= \text{Goto}(I_0, s) \\
&= \{[S \to s \cdot r, \$], \\
&\quad [L \to s\cdot, t/p], \\
&\quad \} \\
I_5 &= \text{Goto}(I_0, a) \\
&= \{[L \to a \cdot Mb, t/p], \\
&\quad [M \to \cdot t, b], \\
&\quad \} \\
I_6 &= \text{Goto}(I_0, t) \\
&= \{[L \to t\cdot, t/p], \\
&\quad \} \\
I_7 &= \text{Goto}(I_2, M) \\
&= \{[S \to LM\cdot, \$], \\
&\quad \} \\
I_8 &= \text{Goto}(I_2, p) \\
&= \{[S \to Lp\cdot, \$], \\
&\quad \}
\end{aligned}
$$

$$
\begin{aligned}
I_9 &= \text{Goto}(I_2, t) \\
&= \{[M \to t\cdot, \$], \\
&\quad \} \\
I_{10} &= \text{Goto}(I_3, L) \\
&= \{[S \to qL \cdot r, \$], \\
&\quad \} \\
I_{11} &= \text{Goto}(I_3, s) \\
&= \{[S \to qs \cdot p, \$], \\
&= \{[L \to s\cdot, r], \\
&\quad \} \\
I_{12} &= \text{Goto}(I_3, a) \\
&= \{[L \to a \cdot Mb, r], \\
&\quad [M \to \cdot t, b], \\
&\quad \} \\
I_{13} &= \text{Goto}(I_3, t) \\
&= \{[L \to t\cdot, r], \\
&\quad \} \\
I_{14} &= \text{Goto}(I_4, r) \\
&= \{[S \to sr\cdot, \$], \\
&\quad \} \\
I_{15} &= \text{Goto}(I_5, M) \\
&= \{[L \to aM \cdot b, t/p], \\
&\quad \} \\
I_{16} &= \text{Goto}(I_5, t) \\
&= \{[M \to t\cdot, b], \\
&\quad \} \\
I_{17} &= \text{Goto}(I_{10}, r) \\
&= \{[S \to qLr\cdot, \$], \\
&\quad \} \\
I_{18} &= \text{Goto}(I_{11}, p) \\
&= \{[S \to qsp\cdot, \$], \\
&\quad \} \\
I_{19} &= \text{Goto}(I_{12}, M) \\
&= \{[L \to aM \cdot b, r], \\
&\quad \} \\
I_{20} &= \text{Goto}(I_{15}, b) \\
&= \{[L \to aMb\cdot, t/p], \\
&\quad \} \\
I_{21} &= \text{Goto}(I_{19}, b) \\
&= \{[L \to aMb\cdot, r], \\
&\quad \} \\
I_{16} &= \text{Goto}(I_{12}, t)
\end{aligned}
$$

Figure 2.3: LR(1) Canonical Collection

The corresponding LALR(1) collection is:

$$
\begin{aligned}
I_0 &= \{[S' \to \cdot S, \$], \\
&\quad [S \to \cdot LM, \$], \\
&\quad [S \to \cdot Lp, \$], \\
&\quad [S \to \cdot qLr, \$], \\
&\quad [S \to \cdot sr, \$], \\
&\quad [S \to \cdot qsp, \$], \\
&\quad [L \to \cdot aMb, t/p], \\
&\quad [L \to \cdot s, t/p], \\
&\quad [L \to \cdot t, t/p], \\
&\quad \} \\
I_1 &= \{[S' \to S\cdot, \$], \\
&\quad \} \\
I_2 &= \{[S \to L \cdot M, \$], \\
&\quad [S \to L \cdot p, \$], \\
&\quad [M \to \cdot t, \$], \\
&\quad \} \\
I_3 &= \{[S \to q \cdot Lr, \$], \\
&\quad [S \to q \cdot sp, \$], \\
&\quad [L \to \cdot aMb, r], \\
&\quad [L \to \cdot s, r], \\
&\quad [L \to \cdot t, r], \\
&\quad \} \\
I_4 &= \{[S \to s \cdot r, \$], \\
&\quad [L \to s\cdot, t/p], \\
&\quad \}
\end{aligned}
\qquad
\begin{aligned}
I_{512} &= \{[L \to a \cdot Mb, t/p/r], \\
&\quad [M \to \cdot t, b], \\
&\quad \} \\
I_{613} &= \{[L \to t\cdot, t/p/r], \\
&\quad \} \\
I_7 &= \{[S \to LM\cdot, \$], \\
&\quad \} \\
I_8 &= \{[S \to Lp\cdot, \$], \\
&\quad \} \\
I_{916} &= \{[M \to t\cdot, \$/b], \\
&\quad \} \\
I_{10} &= \{[S \to qL \cdot r, \$], \\
&\quad \} \\
I_{11} &= \{[S \to qs \cdot p, \$], \\
&= \{[L \to s\cdot, r], \\
&\quad \} \\
I_{14} &= \{[S \to sr\cdot, \$], \\
&\quad \} \\
I_{1519} &= \{[L \to aM \cdot b, t/p/r], \\
&\quad \} \\
I_{17} &= \{[S \to qLr\cdot, \$], \\
&\quad \} \\
I_{18} &= \{[S \to qsp\cdot, \$], \\
&\quad \} \\
I_{2021} &= \{[L \to aMb\cdot, t/p/r], \\
&\quad \}
\end{aligned}
$$

Figure 2.4: LR(1) Canonical Collection

There are no conflicts. Hence the grammar is LALR(1).

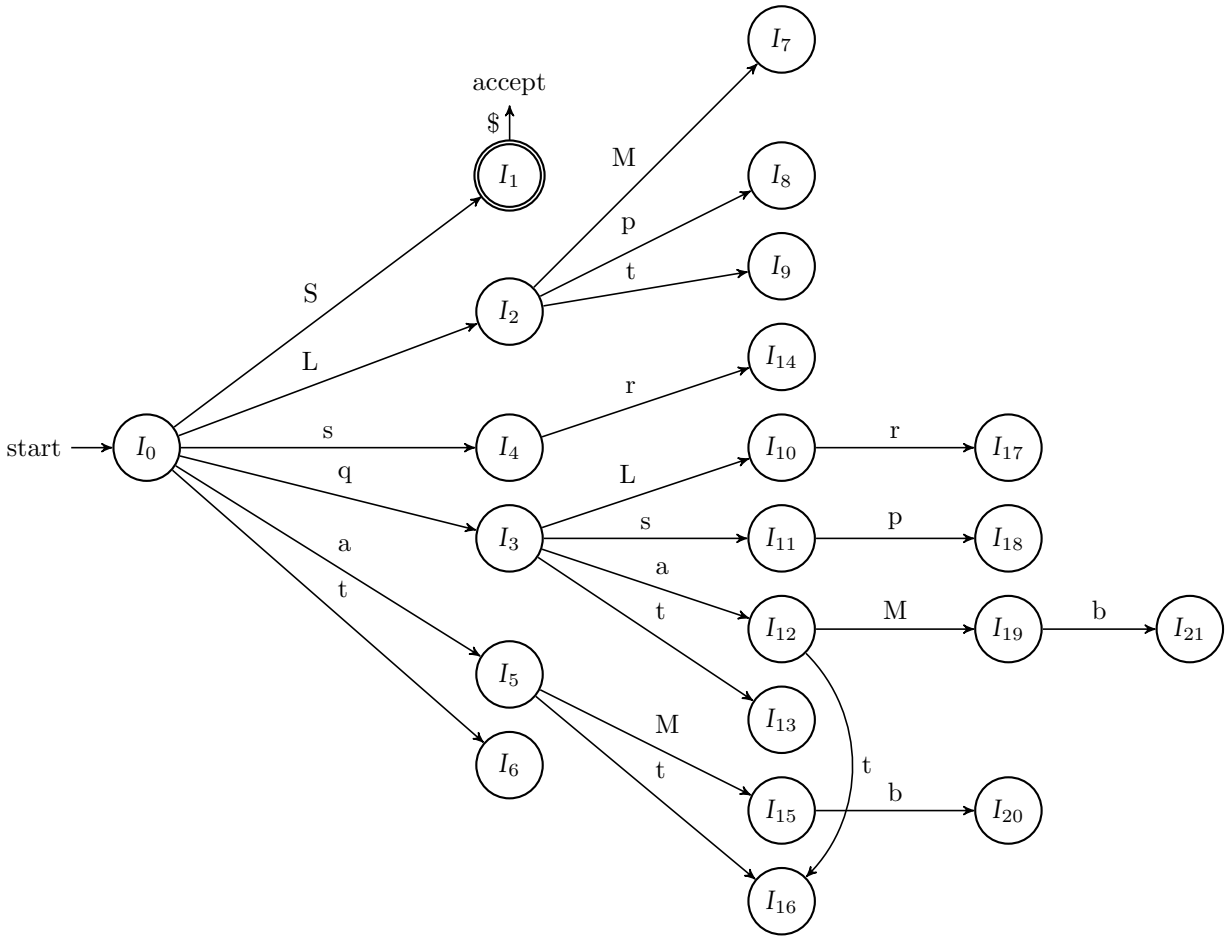

Figure 2.5: LR(0) automaton

STATE	**ACTION**								**GOTO**		
	a	*b*	*p*	*q*	*r*	*s*	*t*	$	S	L	M
0	s512			s3		s4	s613		1	2	
1								**acc**			
2			s8				s916				7
3	s512					s11	s613			10	
4			r7		s14		r7				
512							s916				1519
613			r8		r8		r8				
7								r1			
8								r2			
916		r9						r9			
10					s17						
11			s18		r7						
14								r4			
1519		s2021									
17								r3			
18								r5			
2021								r6			

Table 2.2: LALR(1) Parsing table

Problem 3

3.1 Execution Details

To execute the parser, run the following command:

```
./qrun.sh <relative-path>
```

For example:

```
./qrun.sh ./testcases/t1.quiz
```

Note. *After extracting the zip into the directory, there might be a need to make the attached script executable. Kindly chmod the bash script accordingly.*

3.2 Error Specification

The parser handles very specific error, which are listed as follows

3.2.1 Trailing/Floating close tags

If a question `singleselect` or `multiselect` tag or a `choice` and `correct` tag does not have a opening tag then the parser should yield an error.

For example, consider a statement like `... </singleselect>`, the parser reports an error (replace the ellipsis by `choices` and `correct` tags):

```
line no.: <no.>          No opening tag (singleselect).
```

<no.> is replaced by the appropriate line number.

3.2.2 Unclosed tags

If a question `singleselect` or `multiselect` tag or a `choice` and `correct` tag does not have a opening tag then the parser should yield an error.

For example, consider a statement like `<singleselect marks = "4 "> ...` , the parser reports an error (replace the ellipsis by `choices` and `correct` tags):

```
line no.: <no.>          No closing tag (singleselect).
```

3.2.3 Integral marks

If the `marks` within a question tag is not an integral value, then the parser should report an error.

For example, consider a statement like `<singleselect marks = "4 ab"> ... </singleselect>`, the parser reports an error (replace the ellipsis by `choices` and `correct` tags):

```
line no.: <no.>          Marks should be natural numbers.
```

3.2.4 Marks range

If the `marks` within a question tag is an integral value, but not within the range $[1, 8]$, then the parser should report an error.

For example, consider a statement like `<singleselect marks = "45"> ... </singleselect>`, the parser reports an error (replace the ellipsis by `choices` and `correct` tags):

```
line no.: <no.>          Marks should be in the range 1-8.
```

The parser can also handle the different ranges under which the singleselect and multiselect questions must lie, and gives very specific error messages.

3.2.5 Enclosed in quotations

If the `marks` attribute within a question tag is not assigned to something within double quotes, then the parser should report an error.

For example, consider a statement like `<singleselect marks = 45> ... </singleselect>`, the parser reports an error (replace the ellipsis by `choices` and `correct` tags):

```
line no.: <no.>          Enclose the marks in double quotes.
```

3.2.6 Valid choice and correct tags

If the number of correct and choice tags within a question are *incorrect*, then the parser should report an error. The number of these tags can be correct in the following case:

- If the question if a `singlselect` type, then the number of `correct` tags should be 1 and the number of `choice` tags should be either 3 or 4.
- If the question if a `singlselect` type, then the number of `correct` tags should be less than or equal to the number of `choice` tags and the number of `choice` tags should be either 3 or 4.

Any other case should lead in an error, for example, consider a statement like the following, the parser reports an error

```
<singleselect marks="4">
Question 1
        <choice> 78 </choice>
        <choice> 79 </choice>
        <choice> 80 </choice>
</singleselect>

line no.: <no.>          Error in single select question: in terms of number of choice
and correct tags.
```

Note. *In case of arbitrary inputs, the parser gives undefined outputs. It may produce a generic* `syntax error` *error. Further, the statistics produced by the parser are uptil only the erroneous situations. The parser has the capability of ignoring any stray strings, however please note that any stray* < *or* > *are caught and a* `syntax error` *is reported. This is because the parser deals with brackets judiciously.*