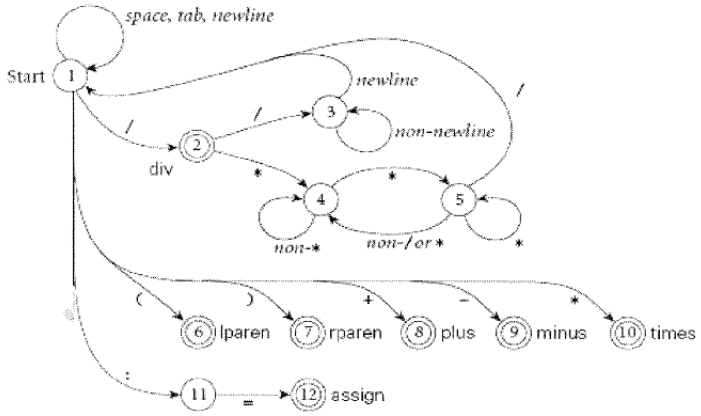
**Concepts of Programming Languages, CSCI 305, Fall 2021**
**Homework #4, complete by Oct. 8**

1. Create a scanning table for the following minimal dfa.




| State | space | tab | newline | / | * | ( | ) | + | - | : | = |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 2 | 10 | 6 | 7 | 8 | 9 | 11 | - |
| **2** | - | - | - | 3 | 4 | - | - | - | - | - | - |
| **3** | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| **4** | 4 | 4 | 4 | 4 | 5 | 4 | 4 | 4 | 4 | 4 | 4 |
| **5** | 4 | 4 | 4 | 1 | 5 | 4 | 4 | 4 | 4 | 4 | 4 |
| **6** | - | - | - | - | - | - | - | - | - | - | - |
| **7** | - | - | - | - | - | - | - | - | - | - | - |
| **8** | - | - | - | - | - | - | - | - | - | - | - |
| **9** | - | - | - | - | - | - | - | - | - | - | - |
| **10** | - | - | - | - | - | - | - | - | - | - | - |
| **11** | - | - | - | - | - | - | - | - | - | - | 12 |
| **12** | - | - | - | - | - | - | - | - | - | - | - |

2. Hand execute the table-driven lexical analyzer pseudo code (Figure 2.11, page 66) on the very simple program.

```
sum :=29.57
write sum+4
```

For each call to the lexical analyzer, list all accesses into the scan_tab, in order as the code is executed. End the call by showing what is returned to the parser.

Each "move" action requires two scan_tab accesses, one to get the action and a second one to get the new_state. Only one access needs to be listed.

First call to the lexical analyzer:
- scan_tab(s, 1)
- scan_tab(u, 16)
- scan_tab(m, 16)
- scan_tab(space, 16)
- return <id, sum>

Next call to the lexical analyzer:
- scan_tab(:, 1)
- scan_tab(=, 11)
- scan_tab(2, 12)
- return <assign, :=>

Next call to the lexical analyzer:
- scan_tab(2, 1)
- scan_tab(9, 14)
- scan_tab(., 14)
- scan_tab(15, 5)
- scan_tab(7, 15)
- scan_tab(newline, 15)
- return <number, 29.57>

Next call to the lexical analyzer:
- scan_tab(newline, 1)
- scan_tab(w, 17)
- scan_tab(w, 1)
- scan_tab(r, 16)
- scan_tab(i, 16)
- scan_tab(t, 16)
- scan_tab(e, 16)
- scan_tab(space, 16)
- return <write, write>

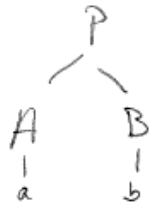
Next call to the lexical analyzer:
- scan_tab(space, 1)
- scan_tab(s, 17)
- scan_tab(s, 1)
- scan_tab(u, 16)
- scan_tab(m, 16)
- scan_tab(+, 16)
- return <id, sum>

Next call to the lexical analyzer:
- scan_tab(+, 1)
- scan_tab(4, 8)
- return <plus, +>

Next call to the lexical analyzer:
- scan_tab(4, 1)
- scan_tab(newline, 14)
- return <number, 4>

Next call to the lexical analyzer:
- scan_tab(newline, 1)
- EOF when in stat

3. Consider the parse tree below.



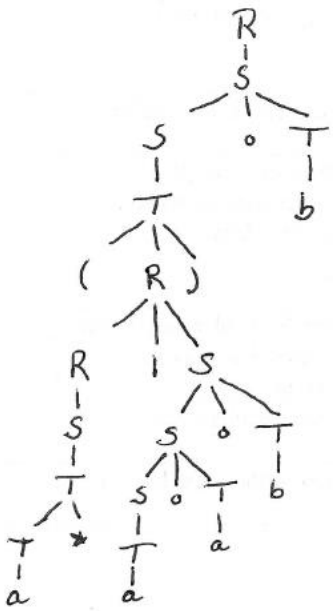

Give a rightmost derivation of the tree:

$P \Rightarrow AB \Rightarrow Ab \Rightarrow ab$

Give a leftmost derivation of the tree:

$P \Rightarrow AB \Rightarrow aB \Rightarrow ab$

4. Consider the parse tree below.




Give a rightmost derivation of the tree:

$R \Rightarrow S \Rightarrow S \circ T \Rightarrow S \circ b \Rightarrow T \circ b \Rightarrow (R) \circ b \Rightarrow (R \mid S) \circ b \Rightarrow (R \mid S \circ T) \circ b \Rightarrow$
$(R \mid S \circ b) \circ b \Rightarrow (R \mid S \circ T \circ b) \circ b \Rightarrow (R \mid S \circ a \circ b) \circ b \Rightarrow (R \mid T \circ a \circ b) \circ b \Rightarrow$
$(R \mid a \circ a \circ b) \circ b \Rightarrow (S \mid a \circ a \circ b) \circ b \Rightarrow (T \mid a \circ a \circ b) \circ b \Rightarrow (T^* \mid a \circ a \circ b) \circ b \Rightarrow$
$(a^* \mid a \circ a \circ b) \circ b \Rightarrow$

5. Create a grammar for the context-free language

$$L = \{x_1 w^R x_2 \# w \mid x_1, x_2, w \in \{a, b\}^* \}$$

where $w^R$ is the reverse of w

| Strings in L | Strings not in L |
|---|---|
| # | ε |
| aaabaa#b | a |
| bbaaaa#abb | b |
| bbabbbb#bbab | bb#aa |
| | ba#ba |
| | aaaaba#bab |

$S \rightarrow X\ P$

$X \rightarrow aX \mid bX \mid \varepsilon$

$P \rightarrow aPa \mid bPb \mid X\#$