Compiler Construction

Sketch parsers based on

(a) recursive descent, and [8 marks]

(b) a table-driven method of your choice (e.g. SLR(1)) [12 marks]

suitable for parsing the following grammar:

\[
\begin{align*}
S & \rightarrow E \text{ eof} \\
E & \rightarrow E + T \mid E - T \mid T \\
T & \rightarrow P \cdot T \mid P \\
P & \rightarrow ( E ) \mid n
\end{align*}
\]

with $S$ as the start symbol. The table-driven parser should include the associated algorithm which interprets the table. The parsers do not need to produce a parse tree, merely to report whether the input string is generated by the above grammar. You may assume there is a routine `lex()` which when called places the next symbol (+, −, ·, (, ), n, eof) in variable `token`. 