Aim:

To design an online Lexical Analyzer simulator to derive leftmost derivation tree and rightmost derivation tree from a given grammar for a goal string.

Programming Language used:

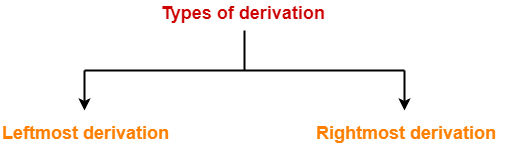
* HTML
* CSS
* JavaScript

Theory:

The process of deriving a string is called as derivation. The geometrical representation of a derivation is called as a parse tree or derivation tree.

Properties Of Parse Tree-

* Root node of a parse tree is the start symbol of the grammar.
* Each leaf node of a parse tree represents a terminal symbol.
* Each interior node of a parse tree represents a non-terminal symbol.



Leftmost Derivation:

The process of deriving a string by expanding the leftmost non-terminal at each step is called as leftmost derivation.

Rightmost Derivation:

The process of deriving a string by expanding the rightmost non-terminal at each step is called as rightmost derivation

Simulator Logic:

1. We start from Starting symbol and represent it as the root node.
2. We check for input symbol and start expanding according to the given grammar.
3. We start elaborating the production i.e., expanding the non-terminals from left to right in case of leftmost derivation tree and from right to left in case of rightmost derivation tree.
4. Similarly, we move the expansion for all the production until we achieve the goal string.

Example:

Let’s consider a Grammar:

S → aB / bA

S → aS / bAA / a

B → bS / aBB / b

Let us consider a string

w = aaabbabbba

Leftmost Derivation**:**

S → a**B**

→ aa**B**B (Using B → aBB)

→ aaa**B**BB (Using B → aBB)

→ aaab**B**B (Using B → b)

→ aaabb**B** (Using B → b)

→ aaabba**B**B (Using B → aBB)

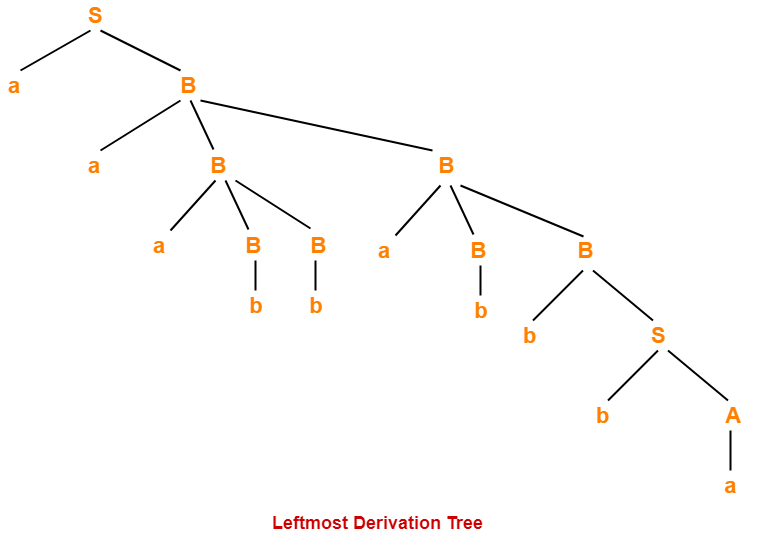
→ aaabbab**B** (Using B → b)

→ aaabbabb**S** (Using B → bS)

→ aaabbabbb**A** (Using S → bA)

→ aaabbabbba (Using A → a)

Leftmost Derivation Tree:



Rightmost Derivation:

S → a**B**

→ aaB**B** (Using B → aBB)

→ aaBaB**B** (Using B → aBB)

→ aaBaBb**S** (Using B → bS)

→ aaBaBbb**A** (Using S → bA)

→ aaBa**B**bba (Using A → a)

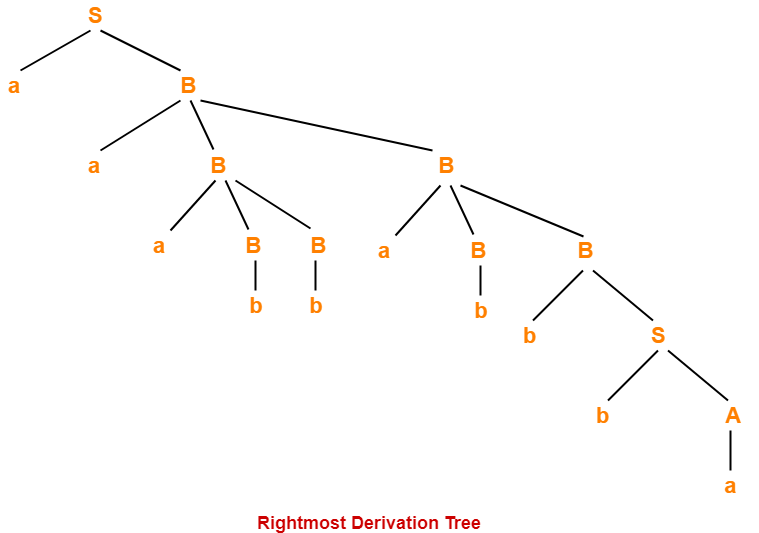
→ aa**B**abbba (Using B → b)

→ aaaB**B**abbba (Using B → aBB)

→ aaa**B**babbba (Using B → b)

→ aaabbabbba (Using B → b)

Rightmost Derivation Tree:



Discussion:

We wrote the online simulator using HTML, CSS, JavaScript and successfully implemented the concept of left and right derivation tree.

Sources of Error:

If the grammar is ambiguous i.e., if there exist more than one left or right derivations for the same string.

For an ambiguous grammar left and right derivation tree of a given grammar may yield different results.