Parsing is used to derive a string using the production rules of a grammar. It is used to check the acceptability of a string. Compiler is used to check whether or not a string is syntactically correct. A parser takes the inputs and builds a parse tree.

A parser can be of two types −

* **Top-Down Parser** − Top-down parsing starts from the top with the start-symbol and derives a string using a parse tree.
* **Bottom-Up Parser** − Bottom-up parsing starts from the bottom with the string and comes to the start symbol using a parse tree.

Design of Top-Down Parser

For top-down parsing, a PDA has the following four types of transitions −

* Pop the non-terminal on the left hand side of the production at the top of the stack and push its right-hand side string.
* If the top symbol of the stack matches with the input symbol being read, pop it.
* Push the start symbol ‘S’ into the stack.
* If the input string is fully read and the stack is empty, go to the final state ‘F’.

Example

Design a top-down parser for the expression "x+y\*z" for the grammar G with the following production rules −

P: S → S+X | X, X → X\*Y | Y, Y → (S) | id

***Solution***

If the PDA is (Q, ∑, S, δ, q0, I, F), then the top-down parsing is −

(x+y\*z, I) ⊢(x +y\*z, SI) ⊢ (x+y\*z, S+XI) ⊢(x+y\*z, X+XI)

⊢(x+y\*z, Y+X I) ⊢(x+y\*z, x+XI) ⊢(+y\*z, +XI) ⊢ (y\*z, XI)

⊢(y\*z, X\*YI) ⊢(y\*z, y\*YI) ⊢(\*z,\*YI) ⊢(z, YI) ⊢(z, zI) ⊢(ε, I)

Design of a Bottom-Up Parser

For bottom-up parsing, a PDA has the following four types of transitions −

* Push the current input symbol into the stack.
* Replace the right-hand side of a production at the top of the stack with its left-hand side.
* If the top of the stack element matches with the current input symbol, pop it.
* If the input string is fully read and only if the start symbol ‘S’ remains in the stack, pop it and go to the final state ‘F’.

Example

Design a top-down parser for the expression "x+y\*z" for the grammar G with the following production rules −

P: S → S+X | X, X → X\*Y | Y, Y → (S) | id

***Solution***

If the PDA is (Q, ∑, S, δ, q0, I, F), then the bottom-up parsing is −

(x+y\*z, I) ⊢ (+y\*z, xI) ⊢ (+y\*z, YI) ⊢ (+y\*z, XI) ⊢ (+y\*z, SI)

⊢(y\*z, +SI) ⊢ (\*z, y+SI) ⊢ (\*z, Y+SI) ⊢ (\*z, X+SI) ⊢ (z, \*X+SI)

⊢ (ε, z\*X+SI) ⊢ (ε, Y\*X+SI) ⊢ (ε, X+SI) ⊢ (ε, SI)