Government College of Engineering, Jalgaon

(An Autonomous Institute of Govt. of Maharashtra)

Name of Student:……………………………………………………
PRN:………. …
Course Teacher: Sharayu Bonde
Date of Performance: **Date of completion:**

Experiment No.: Group B: 05

Title: Implementation of shift reduce parsing algorithm.

Theory:

A shift-reduce parser is a class of efficient, table-driven bottom-up parsing methods for computer languages and other notations formally defined by a grammar. The parsing methods most commonly used for parsing programming languages, LR parsing and its variations, are shift-reduce methodsThe precedence parsers used before the invention of LR parsing are also shift-reduce methods. All shift-reduce parsers have similar outward effects, in the incremental order in which they build a parse tree or call specific output actions

A shift-reduce parser scans and parses the input text in one forward pass over the text, without backing up. (That forward direction is generally left-to-right within a line, and top-to-bottom for multi-line inputs.) The parser builds up the parse tree incrementally, bottom up, and left to right, without guessing or backtracking. At every point in this pass, the parser has accumulated a list of subtrees or phrases of the input text that have been already parsed. Those subtrees are not yet joined together because the parser has not yet reached the right end of the syntax pattern that will combine them.

Shift Reduce Parser in Compiler

Prerequisite – Parsing | (Bottom Up or Shift Reduce Parsers)

Shift Reduce parser attempts for the construction of parse in a similar manner as done in bottom up parsing i.e. the parse tree is constructed from leaves(bottom) to the root(up). A more general form of shift reduce parser is LR parser.

This parser requires some data structures i.e.

A input buffer for storing the input string.

A stack for storing and accessing the production rules.

Basic Operations –

Shift: This involves moving of symbols from input buffer onto the stack.

Reduce: If the handle appears on top of the stack then, its reduction by using appropriate production rule is done i.e. RHS of production rule is popped out of stack and LHS of production rule is pushed onto the stack.

Accept: If only start symbol is present in the stack and the input buffer is empty then, the parsing action is called accept. When accept action is obtained, it is means successful parsing is done.

Error: This is the situation in which the parser can neither perform shift action nor reduce action and not even accept action.

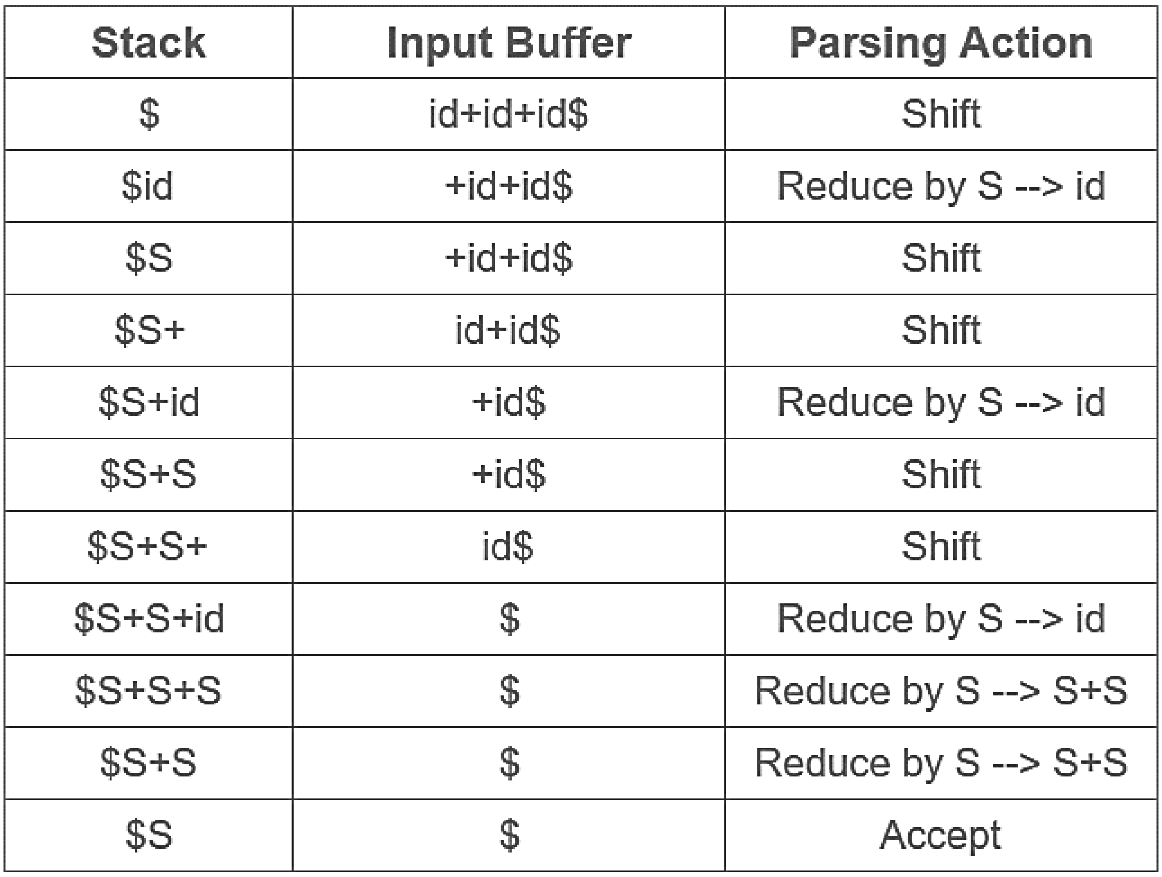
Example 1 – Consider the grammar

S –> S + S

S –> S * S

S –> id

Perform Shift Reduce parsing for input string “id + id + id”.



Stack	Input Buffer	Parsing Action
$	id+id+id$	Shift
$id	+id+id$	Reduce by S --> id
$S	+id+id$	Shift
$S+	id+id$	Shift
$S+id	+id$	Reduce by S --> id
$S+S	+id$	Shift
$S+S+	id$	Shift
$S+S+id	$	Reduce by S --> id
$S+S+S	$	Reduce by S --> S+S
$S+S	$	Reduce by S --> S+S
$S	$	Accept

Result:

Implementation of shift reduce parser has been perform succesfully.

Course Teacher

Sharayu N. Bonde