Name: SWARNAVA CHAKRABORTY
Roll no. 20
Dept. CSE Year. 4th
Subject Code: Compiler Design
Date: 08-01-2021

1) i) b) Bottom Up parser

1) ii) c) the lexical analysis of the program

1) iii) a) LR grammer

1) iv) b) left-most derivation

1) v) c) Canonical LR

(4) part II

Viable Prefix :-

The prefixes of right sentential forms that can appear on the stack of a shift-reduce parser are called viable prefixes.

By definition, a viable prefix is a prefix ~~of~~ a right sentential form that does not continue past the right end of the rightmost handle of that sentential form.

Example :-

Let: $S \rightarrow X_1\ X_2\ X_3\ X_4$

$A \rightarrow X_1\ X_2$

Let $w = X_1\ X_2\ X_3$

Stack	Input
$	$X_1\ X_2\ X_3$
$ X_1	$X_2\ X_3$
$ $X_1\ X_2$	X_3

Stack	Input
$ A	X_3
$ A X_3	$
:	

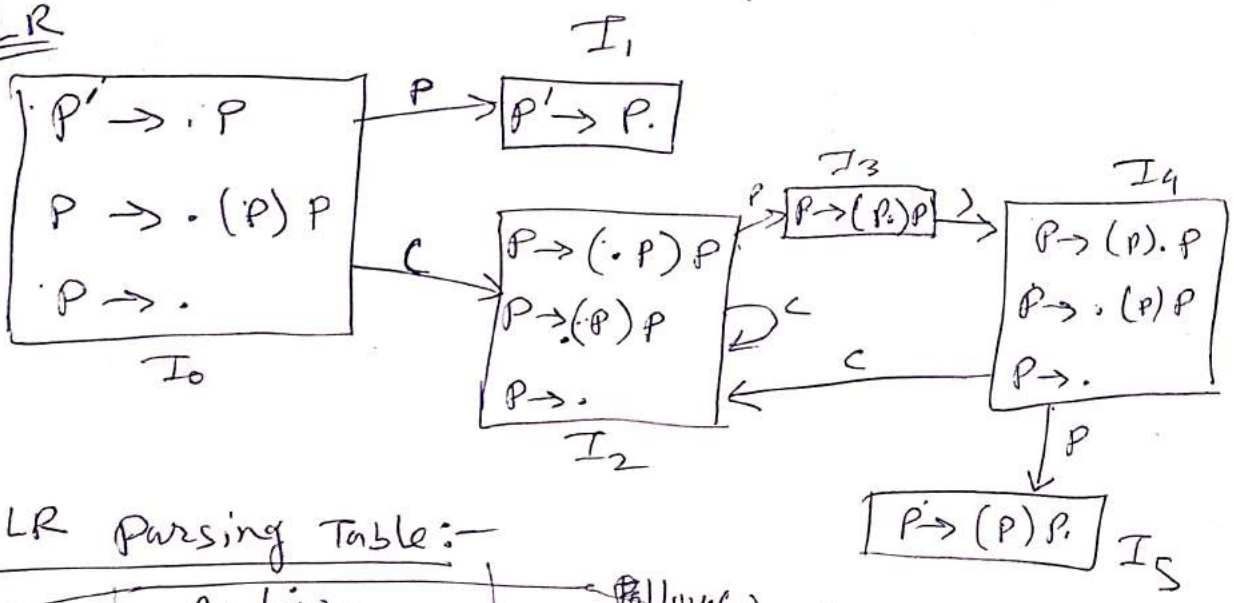
As we see, x_1 x_2 x_3 will never appear on the stack.
So, it is not a viable prefix.

(4) Ans-I

$P \rightarrow P(P)$ ——— (1)

$P \rightarrow \epsilon$ ——— (2)

SLR



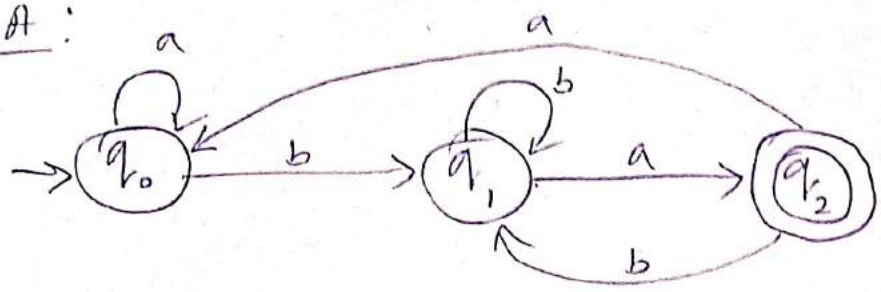
SLR Parsing Table:-

Follow(1) = {$, $}

State	Action			goto
	(	)	$	
0	S_2	r_2	r_2	P
1			Accept	S_1
2	S_2	r_2	r_2	S_3
3		S_4		
4	S_2	r_2	r_2	S_5
5		r_1	r_1	

(2) R.E. : $(a/b)^* ba$.

DFA :



Grammar corresponding to the DFA:

$q_0 \rightarrow a q_0 / b q_1 \quad (q_0, q_1, q_2) \in V$

$q_1 \rightarrow a q_2 / b q_1 \quad (a, b) \in T$

$q_2 \rightarrow a q_0 / b q_1 / \epsilon$

	First	Follow
$q_0 \rightarrow (a q_0 / b q_1)$	$\{a, b\}$	$\{\$\}$
q_1	$\{a, b\}$	$\{\$\}$
q_2	$\{a, b, \epsilon\}$	$\{\$\}$

Last pos

$Last(q_0) = \{\$\}$

$Last(q_1) = \{\$\}$

$Last(q_2) = \{\$\}$

(3) Symbol table :-

Symbol table is an important ds created and maintained by compilers in order to store information about the occurence of various entities such as variable names, function names, objects, classes, interfaces, etc. Symbol table is used by both the analysis and the synthesis parts of a compiler.

A symbol table may serve the following purposes depending upon the language in hand :-

- To store the names of all entities in a structured form at one place.
- To verify if a variable has been declared.
- To implement type checking, by verifying assignments and expressions in the source code are semantically correct.
- To determine the scope of a name.