Name :- Deepak Jaiswal

Sec :- A

Roll No :- 1701010051

Subject :- Compiler Assignment.

* Compiler Design Assignment

Ques-1> Define the term S-R and R-R conflict.

Ans SR Conflict :- In a parsing table, if a cell has both shift move as well as reduce move then shift reduce conflict arises.

RR conflict :- In a parsing table, if a cell has 2 different reduces then reduce - reduce conflict occurs.

Ques.2> Determine whether the following grammar can be parsed by a top down parser or not. In case it can not be top-down parsed, make necessary transformation to that effect.

$E \rightarrow E+T \mid T$
$T \rightarrow T*f \mid f$
$f \rightarrow id \mid (E)$

Soln. The given grammar can't be directly parsed by top down parser because it contains left recursion.

Eliminating left recursion, we get

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \epsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \epsilon$
$F \rightarrow (E) \mid id$

Now we are using predictive parsing technique for solving given grammar.

step-1

Production	First	Follow
E	(, id	$,)
E'	+, ϵ	$,)
T	(, id	+,), $
T'	*, ϵ	+,), $
f	(, id	*, +,), $

Step-2 > Parsing Table

Non-terminal	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to id$			$F \to (E)$		

∴ It is LL(1)

~~Step-3 > Stack Implementation~~

~~Stack~~ ~~Input~~ ~~Production~~

Ques 3> Explain recursive decent parser. Create the parser for the following grammar.

$E \to iE'$

$E' \to +iE' \mid \epsilon$

Soln: Recursive decent parser :-

A parser that uses a set of recursive procedure to recognise its input with no back track is called a Recursive Decent Parsing.

```
main()
{
 E();
 if(l=='$')
    printf("parsing successful");
 }
E()
{ if(l=='i')
   { match('i');
     E'();
} }
```

```
E'()
{ if(l=='+')
    { match('+');
      match('i');
      E'();
    }
  else
      return;
}
```

Ques. 4> Consider the following grammar.

$S \rightarrow AS \mid b$

$A \rightarrow SA \mid a$.

Construct the SLR parse table for grammar. Show the actions of the parser for input string "abab".

Soln.

Step-1 > Augment the given grammar.

$S' \rightarrow S$

$S \rightarrow AS \mid b$

$A \rightarrow SA \mid a$

Wrong.

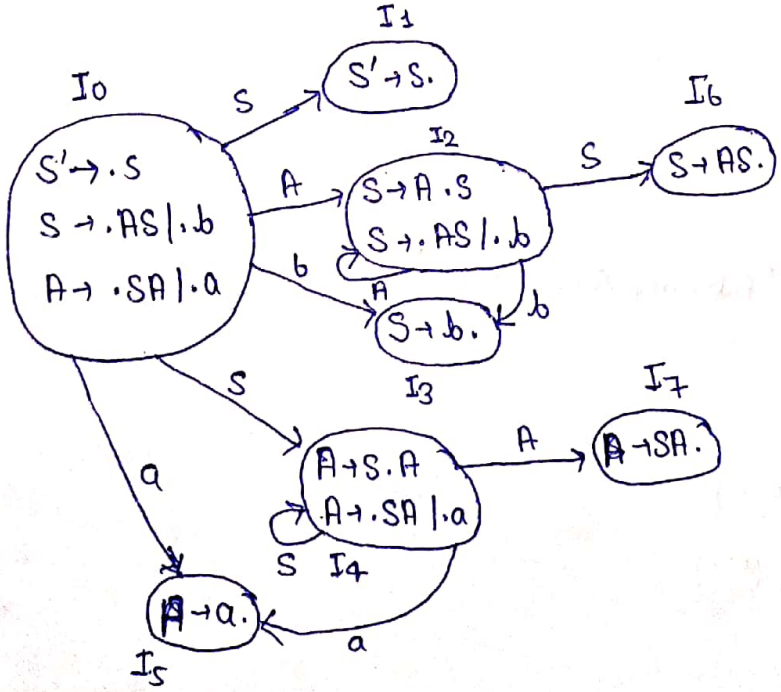
✗ (Doubt while solving this question)

Step-2 > Draw canonical collection.

$S' \rightarrow .S$

$S \rightarrow .AS \mid .b$

$A \rightarrow .SA \mid .a$



Ques:5> Show that the following grammar is LR(1) but not LALR(1).

S → Aa | bAc | Bc | bBa

A → d

B → d

Soln Step-1> Augment the given grammar

S' → S

S → Aa

S → bAc

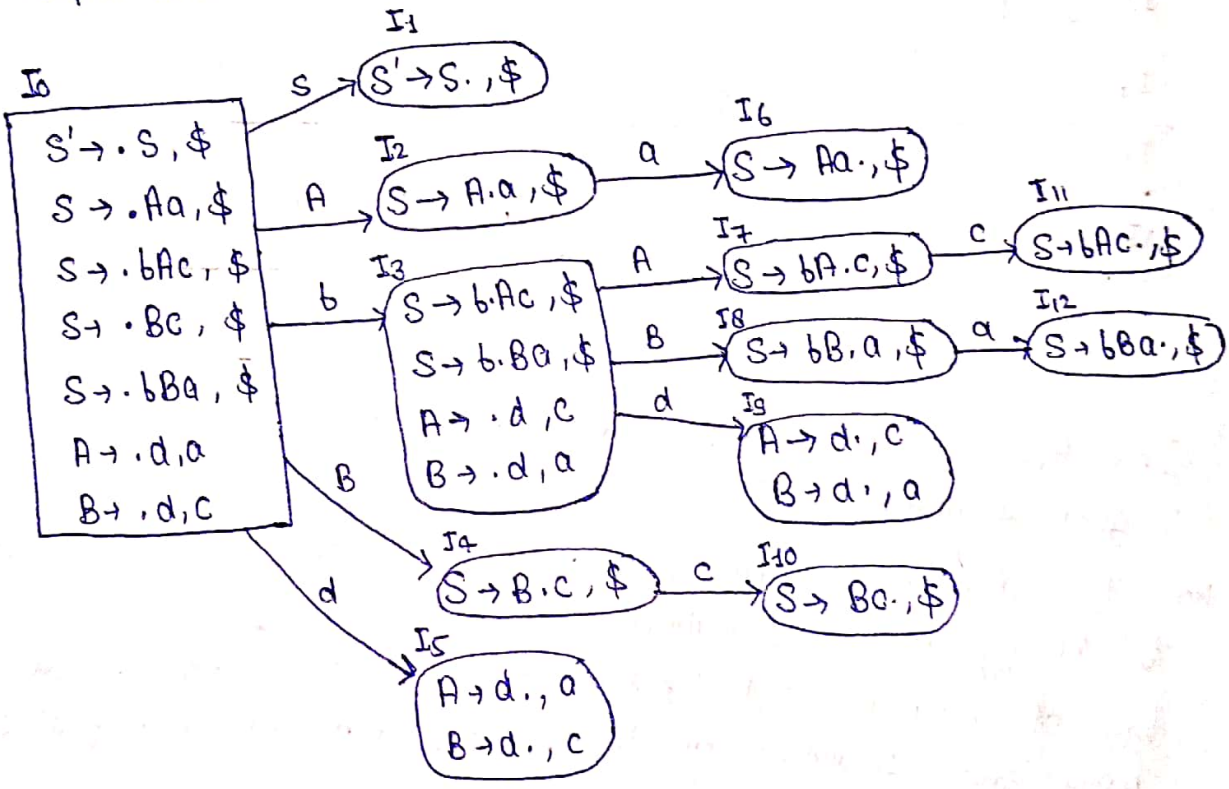
S → Bc

S → bBa

A → d

B → d

Step-2> Draw the canonical collection of LR(1) items.



Step-3 > Number the productions.

S' → S

S → Aa - (1)

S → bAc → (2)

S → Bc - (3)

S → bBa - (4)

A → d - (5)

B → d - (6)

Step 4) Parsing Table

State	Action					Goto		
	a	b	c	d	$	S	A	B
I_0		S_3		S_4		1	2	4
I_1					Accept			
I_2	S_6							
I_3				S_9			7	8
I_4			S_{10}					
I_5	r_5			r_6				
I_6								
I_7			S_{11}					
I_8	S_{10}							
I_9	r_6			r_5				
I_{10}					r_3			
I_{11}					r_2			
I_{12}					r_4			

Since there are no multiple actions in any entry,

∴ given grammar is LR(1).

Now, for LALR(1), we have to find those productions which are same but having different lookaheads.

When obtaining the LALR(1) parsing table by merging states, we will merge states I_5 and I_9, and the resulting state will be as follows:

$I_5 + I_9 = I_{59}$: $A \to d.$, a/c

$B \to d.$, a/c

It is basically R-R conflict.

∴ given grammar is not LALR(1).

Ques.6> What is precedence function? Consider the following operator precedence matrix draw precedence graph and compute the precedence function.

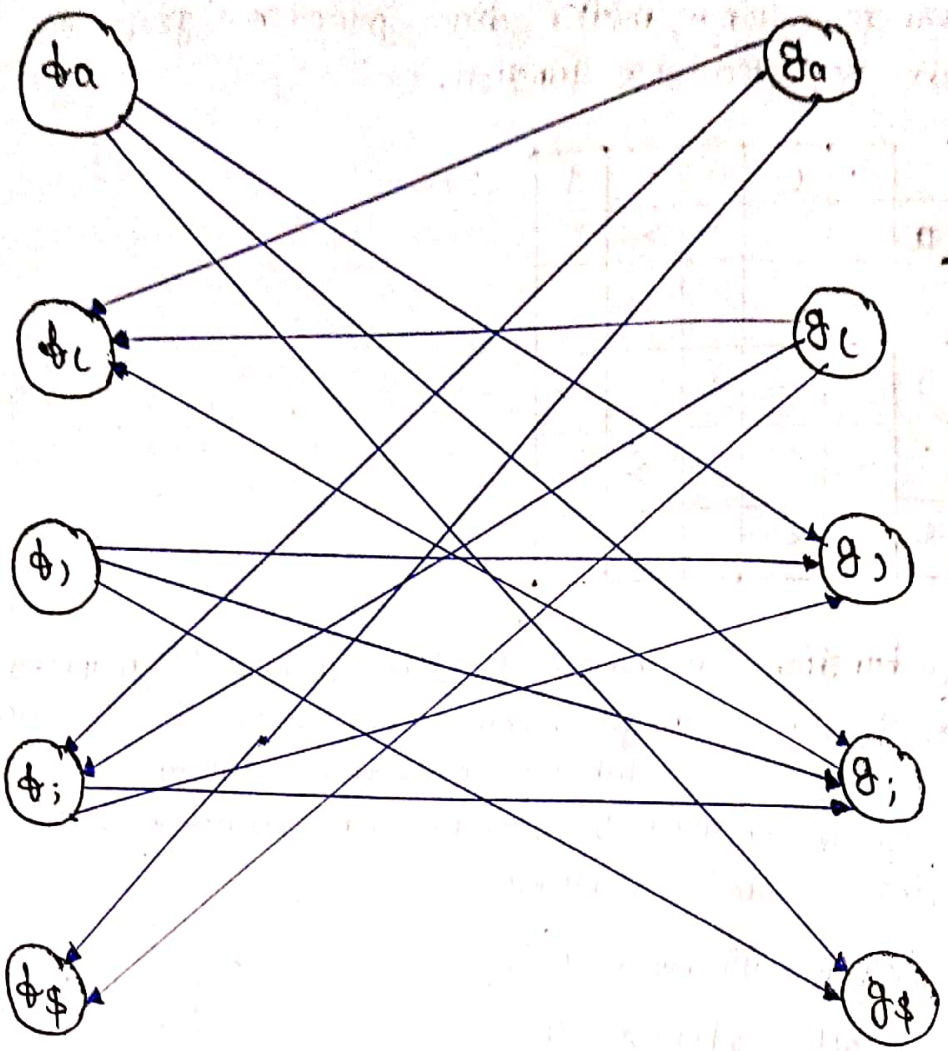
	a	(	)	;	$
a			>	>	>
(	<	<	=	<	
)			>	>	>
;	<	<	>	>	
$	<	<			

Ans. **Precedence Function:** Precedence functions perform the mapping of terminal symbols to the integers. Compilers using operator-precedence parser need not store the table of precedence relation. The table can be encoded by two precedence function ϕ and g that map terminal symbol to integers.

1. $\phi(a) < g(b)$ whenever $a < b$
2. $\phi(a) = g(b)$ whenever $a = b$
3. $\phi(a) > g(b)$ whenever $a > b$

P.T.O.

Precedence graph:



Precedence function:

	a	(	)	;	$
f	1	0	2	2	0
g	3	3	0	1	0

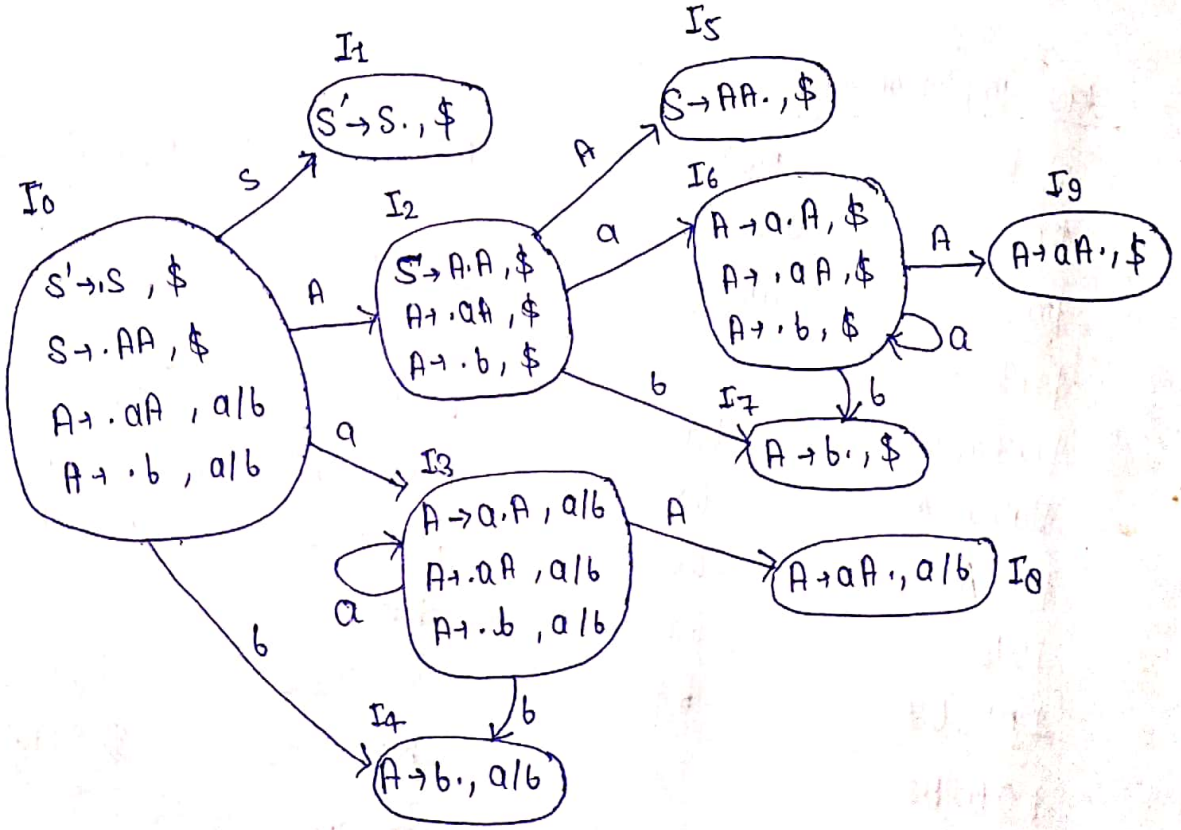
Ques-7> Construct the CLR parsing table for the following grammar and parse the string "aabb". Show each and every step of algorithm.

$S \to AA$
$A \to aA$
$A \to b$

Soln. Step-1> Augment the given grammar

$S' \to S$
$S \to AA$
$A \to aA$
$A \to b$

Step-2> Draw canonical collection of CLR(1) item:



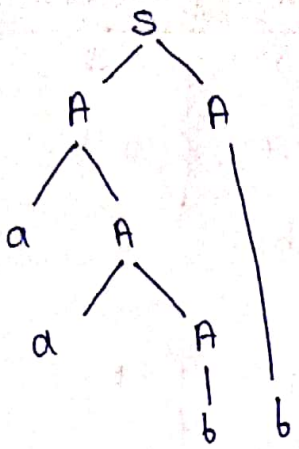
CLR(1) table:

State	Action			Goto	
	a	b	$	S	A
0	S_3	S_4		1	2
1			Accept		
2	S_6	S_7			5
3	S_3	S_4			8
4	r_3	r_3			
5			r_1		
6	S_6	S_7			9
7			r_3		
8	r_2	r_2			
9			r_2		

Stack Implementation:

Stack	Input	Action	Production Used
$0	aabb$	Shift S_3	-
$0a3	abb$	Shift S_3	-
$0a3a3	bb$	Shift S_4	
$0a3a3b4	b$	Reduce r_3	A → b
$0a3a3A8	b$	Reduce r_2	A → aA
$0a3A8	b$	Reduce r_2	A → aA
$0A2	b$	Shift S_7	-
$0A2b7	$	Reduce r_3	A → b
$0A2A5	$	Reduce r_1	S → AA
$0S1	$	Accept.	-

Parse Tree.



Ques. 8) Give algorithm for constructing of predictive parsing table. Consider the grammar and construct predictive parsing table.

$S \rightarrow iEtSS_1 \mid a$

$S_1 \rightarrow eS \mid E$

$E \rightarrow b$

Soln. Algorithm:

```
repeat
    let X be the top stack symbol and 'a' the symbol pointed by pointer
    if X is a terminal or $, then
        if X = a then
            pop X from the stack and move pointer forward
        else
            error()
    else
        if M[X,a] = X → Y1 Y2 ..... Yk then begin
            pop X from the stack
            push Yk, Yk-1 ..... Y1 onto the stack with Y1 on top
            output the production X → Y1 Y2 ..... Yk
        else
            error();
Until X = $
```

Grammar:

$S \rightarrow iEtSS_1 \mid a$

$S_1 \rightarrow eS \mid \epsilon$

$E \rightarrow b$

	FIRST	FOLLOW
S	i, a	e, $
S_1	e, ϵ	e, $
E	b	t

Parsing Table.

	a	b	e	i	t	$
S	$S \rightarrow a$			$S \rightarrow iEtSS_1$		
S_1			$S_1 \rightarrow eS$ $S_1 \rightarrow \epsilon$			$S_1 \rightarrow \epsilon$
E		$E \rightarrow b$				

∵ Table contain multiple entries.

∴ given grammar is ambiguous.

So, it is not LL(1).

Ques-9) State the problem associated with top down parsing.

Ans. Following are the problems associated with top down parsing

→ Backtracking

→ Left Recursion

→ Left factoring

→ Ambiguity