Class Test #2 (Set-A)
CSTE-4105 (Compiler Construction)
Date: 06/06/2024

Answer the following questions: (Time: 45 minutes)

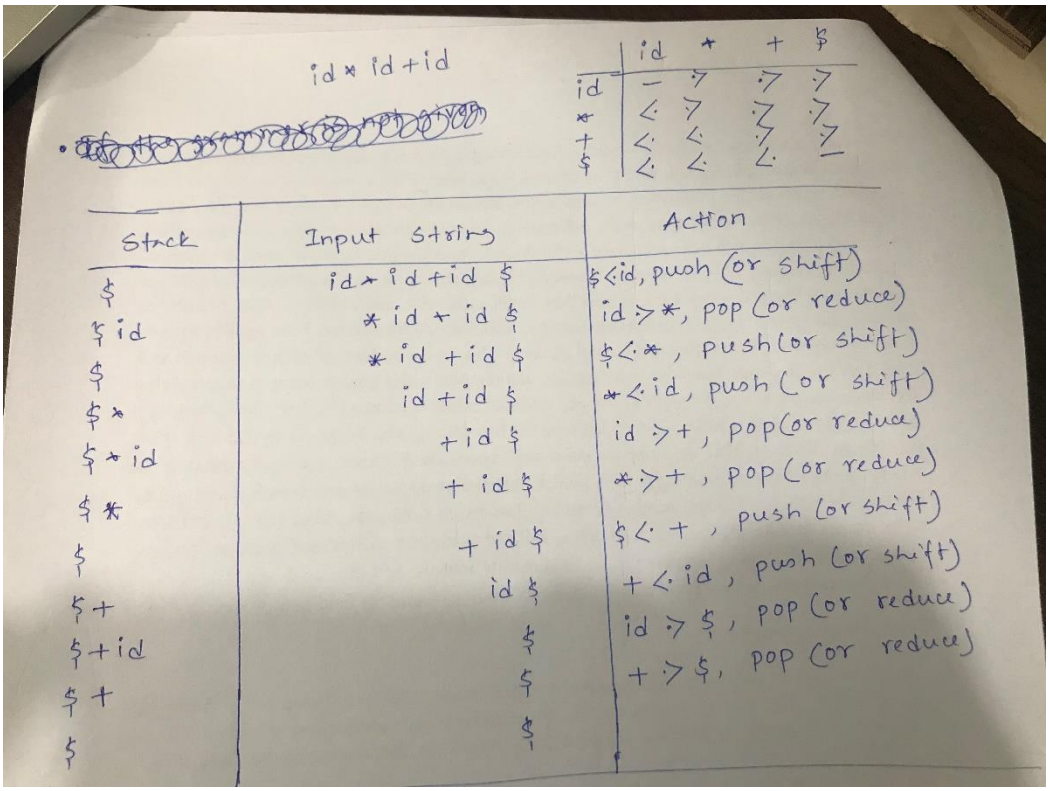
1. Write down the conditions for a grammar to be an operator grammar. Convert the following grammar into operator grammar: **2+3=5**

P → SR|S
R → bSR|bS
S → WbS|W
W → L*W|L
L → id

Answer provided in the slide.

2. Show the parsing steps for the string ***id * id + id*** using operator precedence parsing technique (Note: You must include the precedence table in your answer). **6**

Answer:



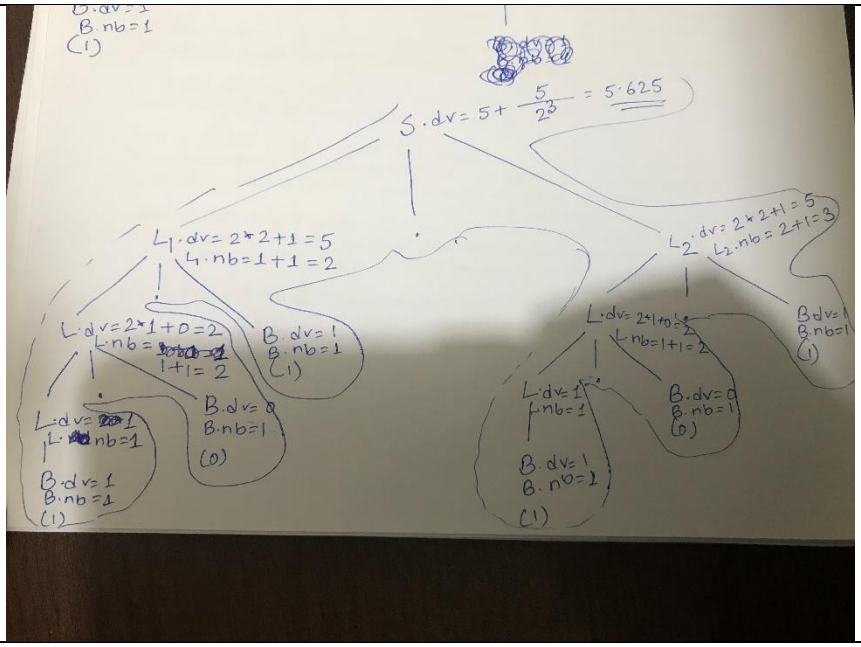
3. What do you mean by Shift-Reduce conflict? Explain with an example. **5**

See slides for the answer.

4. **5**

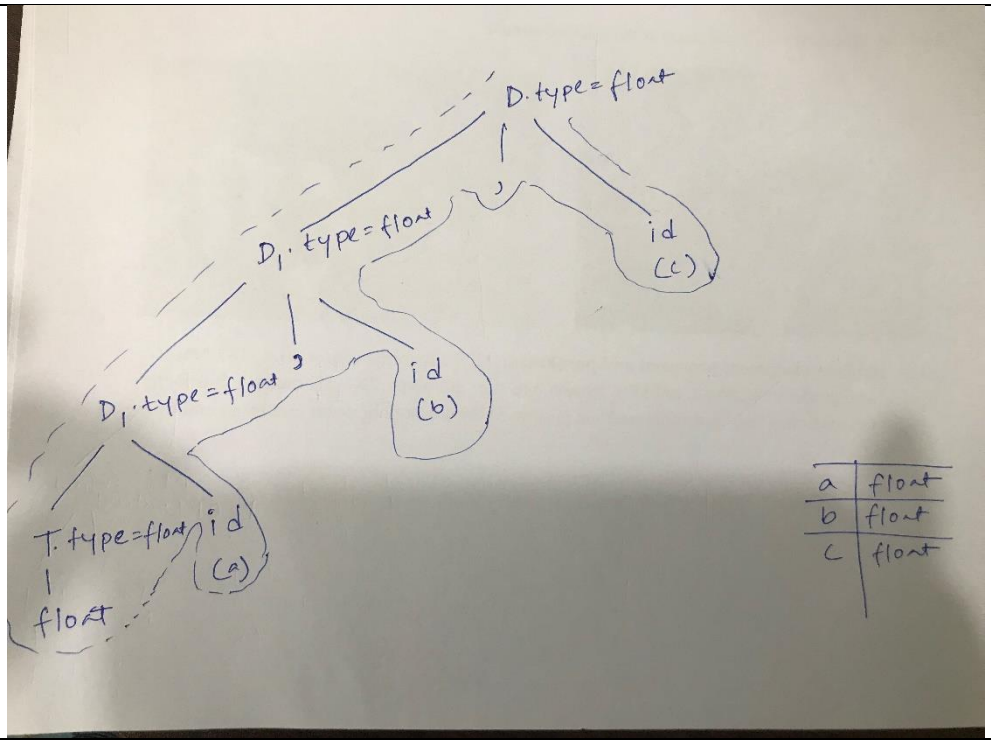
Production	Semantic Rule
$S \rightarrow L_1.L_2$	$\{S.dv = L_1.dv + \frac{L_2.dv}{2^{L_2.nb}}\}$
L → L.B	{L.dv = 2*L.dv+B.dv L.nb = L.nb+B.nb}
L → B	{L.dv = B.dv L.nb = B.nb}
B → 0	{B.dv = 0 B.nb = 1}
B → 1	{B.dv = 1 B.nb = 1}

Show an annotate parse tree for the input expression **101.101** according to the following syntax-directed definition that converts binary to decimal with fraction:

		
5.	Contrast quadruples and triples with an example. See slides for the answer.	**4**

Class Test #2 (Set-B)
CSTE-4105 (Compiler Construction)
Date: 06/06/2024

Answer the following questions: (Time: 45 minutes)		
1.	Differentiate among LR parsers. See slides for the answer.	**4**
2.	What is precedence function? Show how to construct precedence function with an example. See slides for the answer.	**1+4=5**
3.	What do you mean by S-attributed and L-attributed SDD? Explain with examples. See slides for the answer.	**5**
4.	"Dependency graph should not contain any cycle"-why? Show an annotate parse tree for the input expression ***float a, b, c*** according to the following syntax-directed definition that stores type information into symbol table: Production | Semantic Rule $D \rightarrow D_1$, **id** | *{Addtype(id, D_1.type) D.type=D_1.type}* $D \rightarrow T$ **id** | *{Addtype(id, T.type) D.type=T.type}* $T \rightarrow$ *int* | *T.type =int* $T \rightarrow$ *char* | *T.type =char* $T \rightarrow$ float | *T.type =float* Answer: Part 1: See slides for the answer. Part 2:	**1+4=5**



5. What is backpatching? Write down three-address code for the following segment of C code: **1+5=6**

```
c = 0
do
{
if (a < b) then
x++;
else
x–;
c++;
} while (c < 5)
```

Answer:

1. ***c = 0***
2. ***if (a < b) goto 4***
3. ***goto 7***
4. ***T1 = x + 1***
5. ***x = T1***
6. ***goto 9***
7. ***T2 = x – 1***
8. ***x = T2***
9. ***T3 = c + 1***
10. ***c = T3***
11. ***if (c < 5) goto 2***
12. ***

Class Test #2 (Set-C)
CSTE-4105 (Compiler Construction)
Date: 06/06/2024

Answer the following questions: (Time: 45 minutes)

1.	Define handle with an example. What are the rules for constructing *closure* of item sets and *goto* operation? See slides for the answer.	**2+3=5**
2.	Show how to parse the input ***a * b + a*** using the following grammar and the parsing table: E → E+T | T T → TF | F F → F* | a | b	**7**

LR Parsing Table

State	Action					goto		
	+	*	a	b	$	E	T	F
0			s4	s5		1	2	3
1	s6				accept			
2	r2		s4	s5	r2			7
3	r4	s8	r4	r4	r4			
4	r6	r6	r6	r6	r6			
5	r6	r6	r6	r6	r6			
6			s4	s5			9	3
7	r3	s8	r3	r3	r3			
8	r5	r5	r5	r5	r5			
9	r1		s4	s5	r1			7

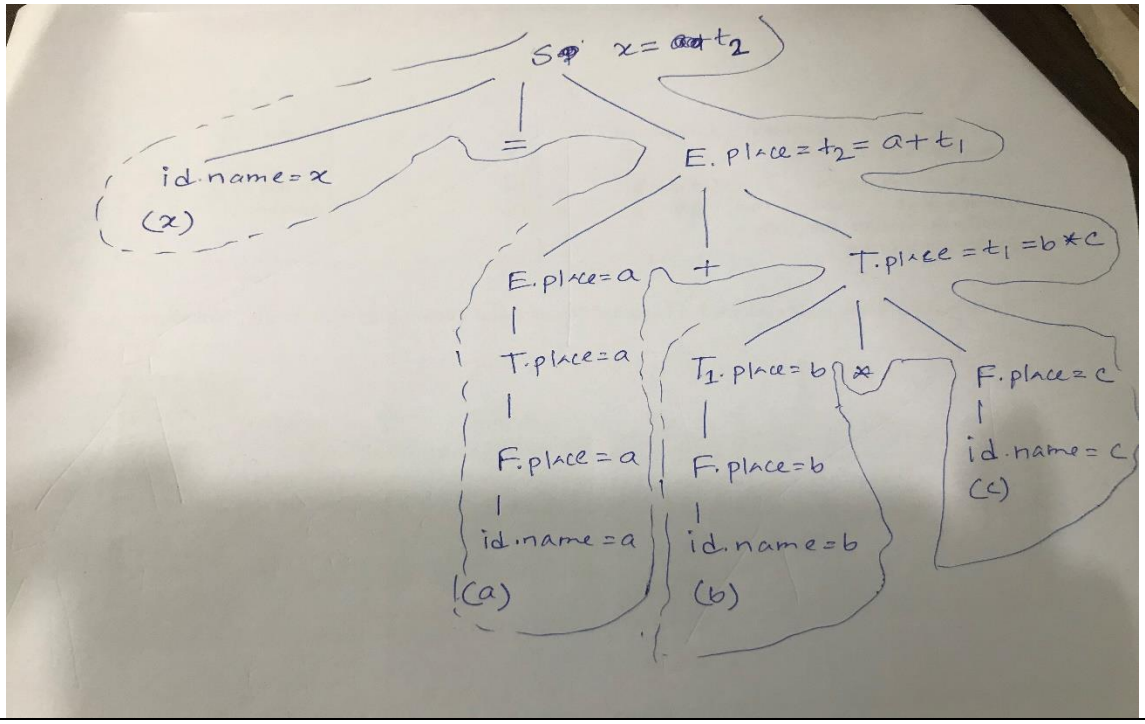
Answer:

Stack	Input String	Action
0	a * b + a $	Shift
0 a 4	* b + a $	Reduce by F → a.
0 F 3	* b + a $	Shift
0 F 3 * 8	b + a $	Reduce by F → F *
0 F 3	b + a $	Reduce by T → F
0 T 2	b + a $	Shift
0 T 2 b 5	+a $	Reduce by F → b
0 T 2 F 7	+a $	Reduce by T → TF
0 T 2	+a $	Reduce by E → T
0 E 1	+a $	Shift
0 E 1 + 6	a$	Shift
0 E 1 + 6 a 4	$	Reduce by F → a
0 E 1 + 6 F 3	$	Reduce by T → F
0 E 1 + 6 T 9	$	Reduce by E → E + T
0 E 1	$	Accept

3.	What do you mean by Reduce-Reduce conflict? Explain with an example. See slides for the answer.	**5**
4.	Show an annotate parse tree for the input expression ***x = a + b * c*** according to the following syntax-directed definition that generates three address code:	**5**

Production	Semantic Rule
$S \rightarrow$ ***id***$=E$	*{gen(id.name, E.place);}*
$E \rightarrow E+T$	*{E.place=newtemp();* *gen(E.place=E.place+T.place);}*
$E \rightarrow$ T	*{ E.place=T.place}*
$T \rightarrow T_1*F$	*{T.place=newtemp();* *gen(T.place=T_1.place*F.place);}*
$T \rightarrow F$	*{ T.place=F.place}*
$F \rightarrow$ **id**	*{ F.place=id.name}*

Answer:



5.	Define indirect triple with an example. See slides for the answer.	**3**