12/10/2022
WEDNESDAY

CS6109 - TUTORIAL II

NAME: ASHWIN MUTHURAMAN A

REGISTER NUMBER: 2020103005

SUBJECT CODE: CS6109

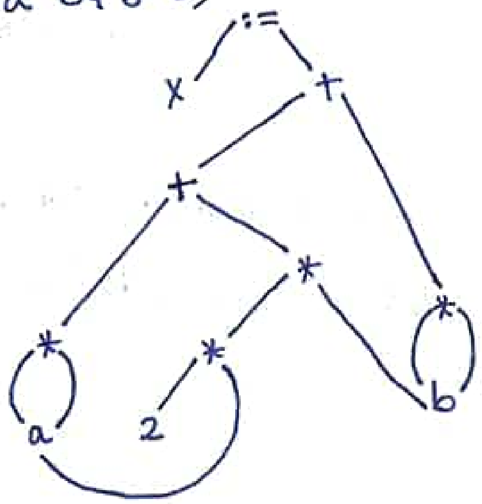
SUBJECT TITLE: COMPILER DESIGN

1) Given the code fragment, draw the DAG.

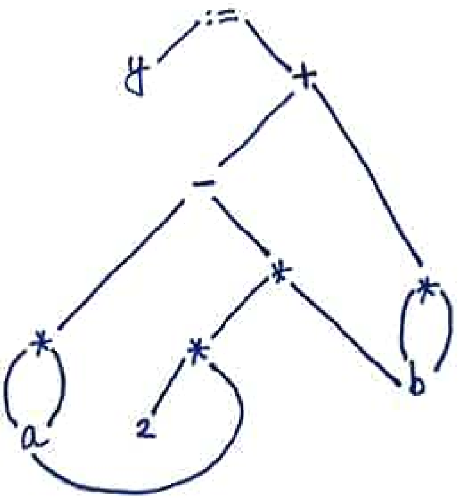
$x := a*a+2*a*b+b*b;$

$y := a*a-2*a*b+b*b;$

Ans: (i) $x := a*a+2*a*b+b*b;$



(ii) $y := a*a-2*a*b+b*b;$



2) Draw the annotated Parse Tree for the following:

$P \rightarrow D S \quad \{S.dl = D.dl\}$

$D_1 \rightarrow var\ V;\ D_2 \quad \{D_1.dl = addList(V.name, D_2.dl)\}$

$\mid \epsilon \quad \{D_1.dl = NULL\}$

$S_1 \rightarrow V := E;\ S_2 \quad \{check(V.name, S_1.dl);\ S_2.dl = S_1.dl\}$

$\mid \epsilon$

$V \rightarrow x \quad \{V.name = 'x'\}$

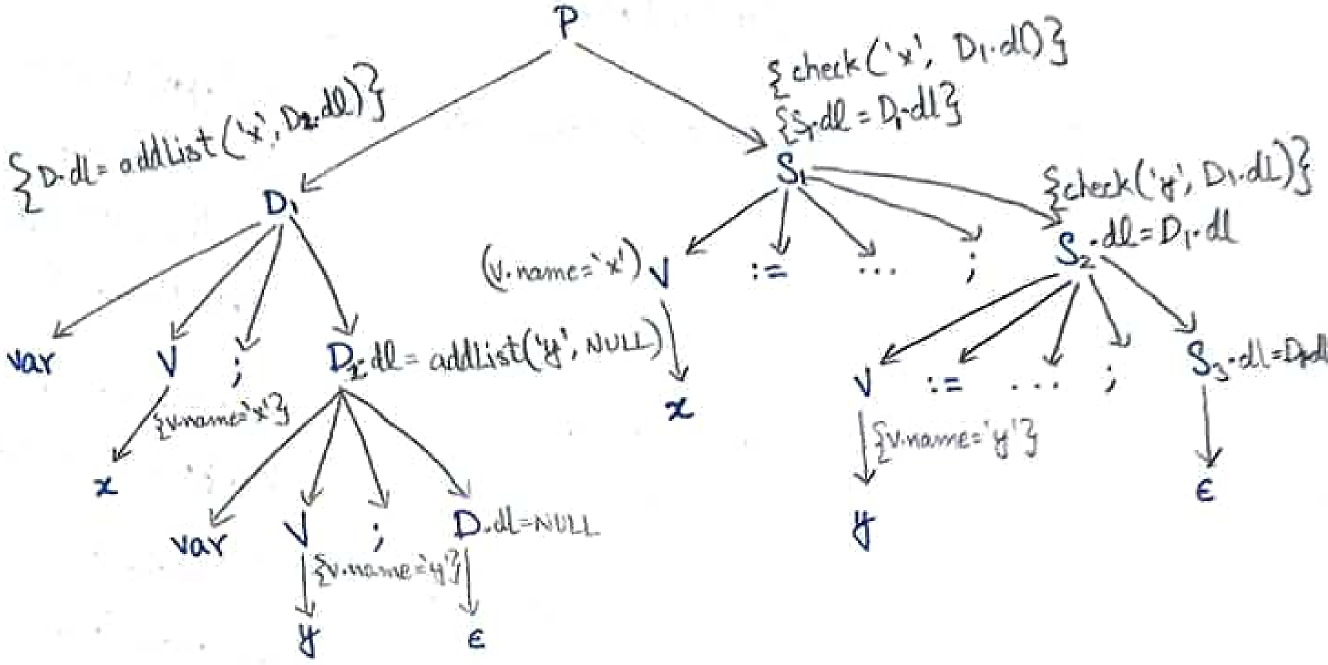
$\mid y \quad \{V.name = 'y'\}$

$\mid z \quad \{V.name = 'z'\}$

Draw the AST| Annotated Parse Tree to the following code, what would the attribute structure look like?

```
var x;
var y;
x:= ...;
y:= ...;
```

Ans: The annotated Parse Tree is as follows:



3) Add rules to compute the decimal value of a signed binary number and evaluate for "-100".

Productions:

$\text{Number} \rightarrow \text{Sign List}$

$\text{Sign} \rightarrow +$
$\mid -$

$\text{List}_0 \rightarrow \text{List}_1 \text{ Bit}$
$\mid \text{Bit}$

$\text{Bit} \rightarrow 0$
$\mid 1$

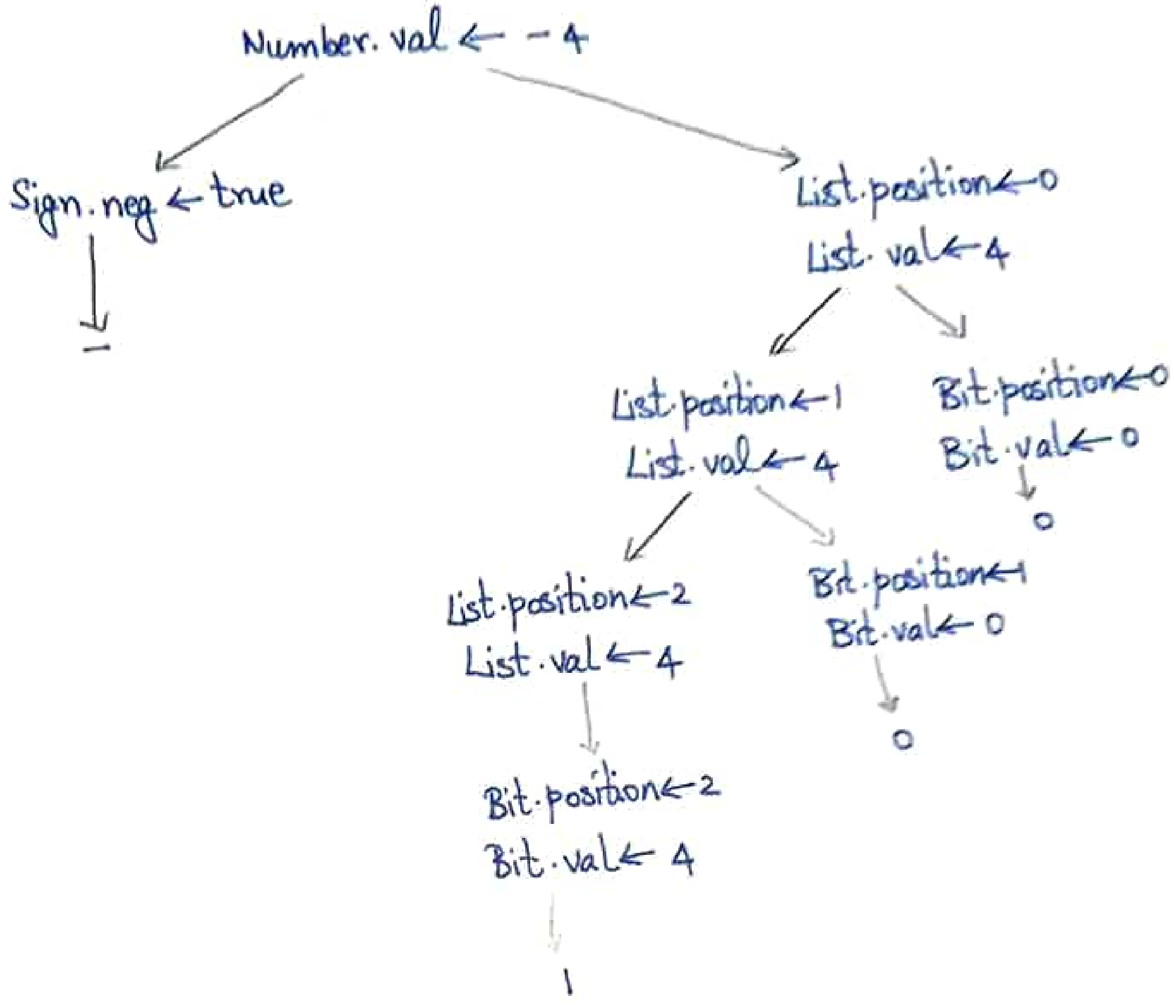
Ans: The attribute rule will be:

Production	Attribute Rules

Number → Sign List	$List.position \leftarrow 0$
	if $Sign.neg$
	then $Number.val \leftarrow -List.va$
	else $Number.val \leftarrow List.val$

Sign → +	$Sign.neg \leftarrow false$
| −	$Sign.neg \leftarrow true$
$List_0 \rightarrow List_1 \; Bit$	$List_1.position \leftarrow List_0.position+1$ $Bit.position \leftarrow List_0.position$ $List_0.val \leftarrow List_1.val + Bit.val$
| Bit	$Bit.position \leftarrow List_0.position$ $List_0.val \leftarrow Bit.val$
Bit → 0	$Bit.val \leftarrow 0$
| 1	$Bit.val \leftarrow 2^{Bit.position}$.

The evaluation for "-100" is as follows:


Number. val ← −4
Sign. neg ← true
−
List. position ← 0
List. val ← 4
List. position ← 1
List. val ← 4
Bit. position ← 0
Bit. val ← 0
0
List. position ← 2
List. val ← 4
Bit. position ← 1
Bit. val ← 0
0
Bit. position ← 2
Bit. val ← 4
1