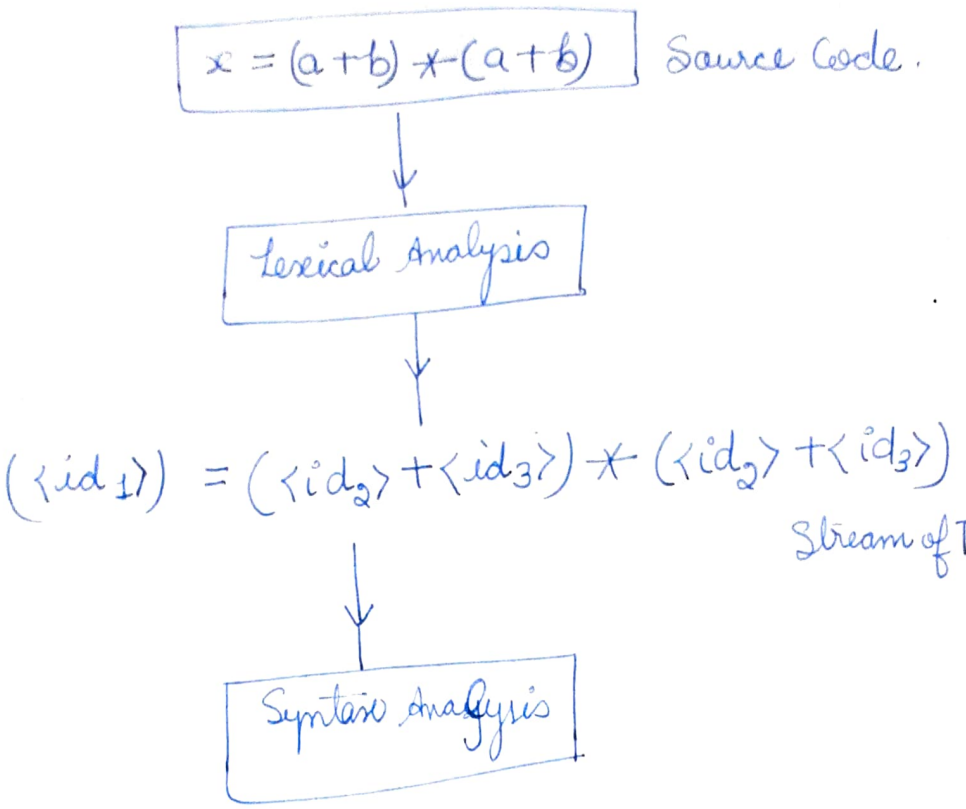
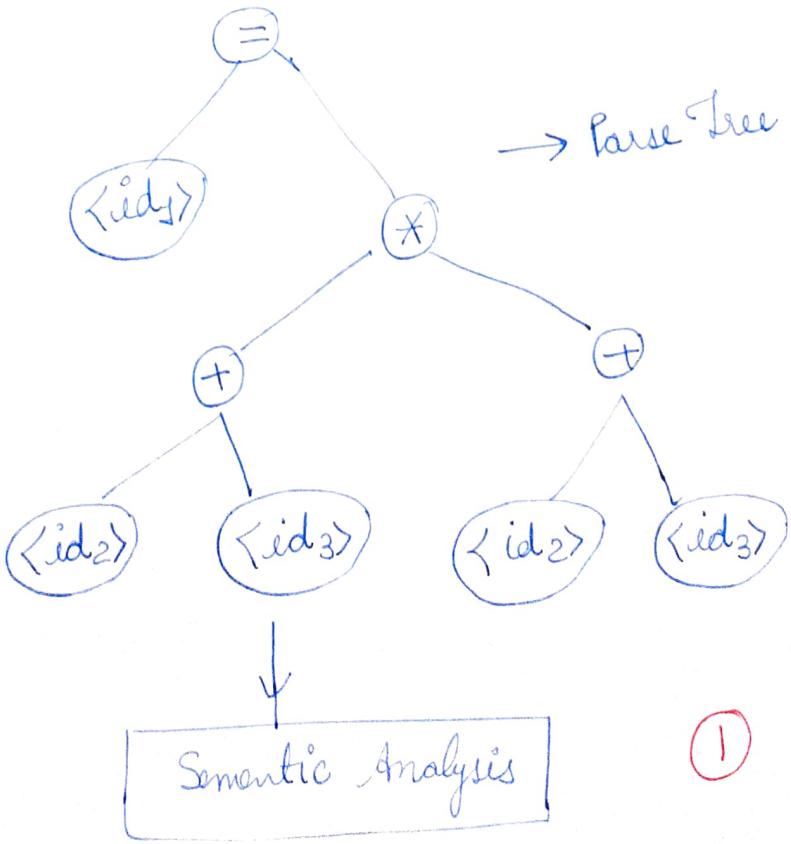
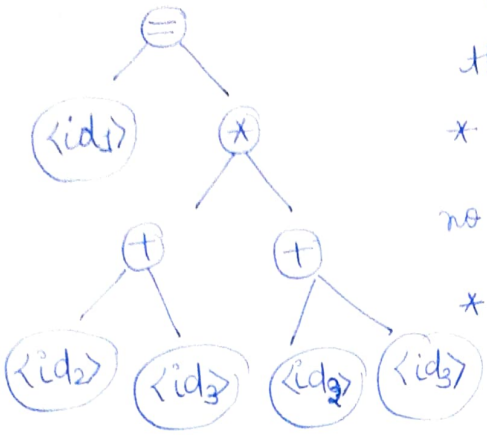
Ques-1 Describe each phases of compiler also construct output of each phase of the following input string

$x = (a+b) * (a+b)$





Correct Parse Tree



* if there is any type conversion. It's perform these conversion.
* In this no constant, so, no need any conversion
* If constant, convert some folting point values.

↓

~~Integrated~~ Intermediate Code Generation

$t_1 = id_2$
$t_2 = id_3$
$t_3 = t_1 + t_2$
$t_4 = t_3 * t_3$
$id_1 = t_4$

→ Intermiate code in the form of 3-address code

↓

Code optimization

$t_1 = id_2 + id_3$
$id_1 = t_1 * t_1$

optimised IC in form of 3-address code.

* CPU can executed faster.

↓

Code Generation

```
LODA R1, id2
LODA R2, id3
ADD  R1, R2
MUL  R1, R1
```

```
STORE id1, R1
```

Relatively ALL

Ques-2 Describe each phases of compiler also construct output of each phase of the following input string:

1. $x = (a+b) * (c+d)$
2. position = initial + rate * 60
3. $a = (b+c) * (b+c) * 2$

(1) $x = (a+b) * (c+d)$

construct all phases

$x = (a+b) * (c+d)$

↓

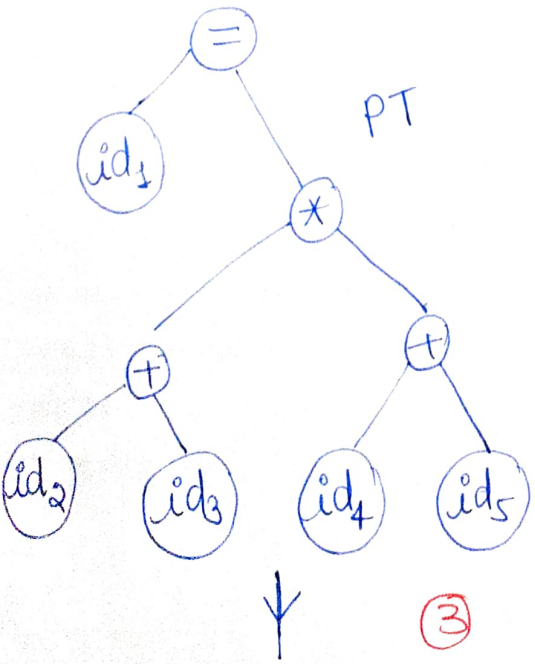
LA → Tokens

$(\langle id_1 \rangle) = (\langle id_2 \rangle + \langle id_3 \rangle) * (\langle id_4 \rangle + \langle id_5 \rangle)$

↓

Syntax Analysis

↓



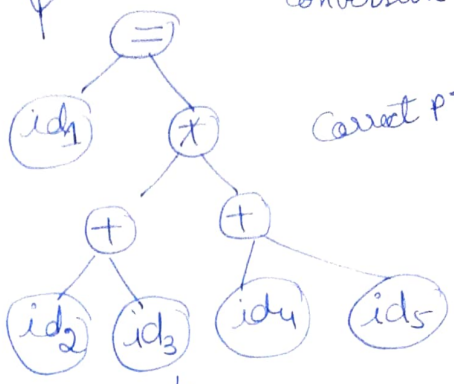
PT

↓

(3)

Semantic Analysis → There is no constant so, no need any conversion.

↓



Correct PT

↓

Intermediate code Generation

↓

IC 3-Address code

$t_1 = id_2 + id_3$

$t_2 = id_4 + id_5$

$t_3 = t_1 * t_2$

$id_1 = t_3$

↓

Code Optimization

↓

optimised IC 3-Address Code

$t_1 = id_2 + id_3$

$t_2 = id_4 + id_5$

$id_1 = t_1 * t_2$

↓

Code Generation

↓

```
LOAD R1, id2
LOAD R2, id3
ADD R1, R2
LOAD R3, id4
LOAD R4, id5
```

(4)

ADD R_3, R_4

MUL R_1, R_3

STORE id_1, R_1

Relatively target code

Target code Long is ALL.

---

(2) position = initial + rate * 60

construct all phases.

position = ~~rate~~ initial + rate * 60

↓

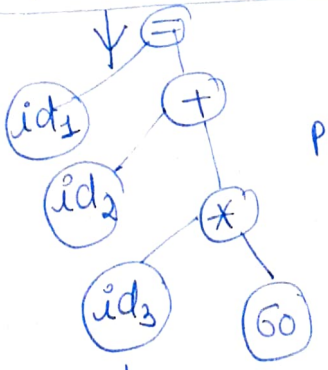
Lexical Analysis

↓

$\langle id_1 \rangle = \langle id_2 \rangle + \langle id_3 \rangle * 60$ Tokens

↓

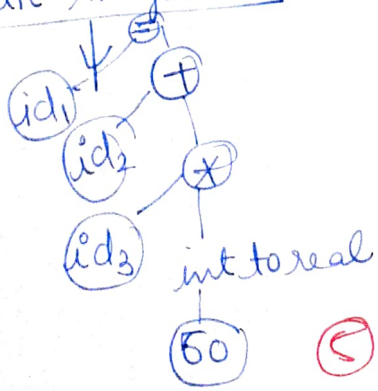
Syntax Analysis



PT

↓

Semantic Analysis



Correct PT

(5)

↓

Intermediate Code Generation

$t_1 = int\ to\ real(60)$

$t_2 = id_3$

$t_3 = t_1 * t_2$

$t_4 = id_4$

$t_5 = t_3 + t_4$

$id_1 = t_5$

IC 3-Address code

↓

Code Generation

↓

$t_1 = id_3 * 60.0$

$id_1 = id_2 + t_1$

optimized 3-Address code

↓

Code Generation

↓

LOAD R_1, id_3

MUL $R_1, 60.0$

LOAD R_2, id_2 — Relatively TC

ADD R_1, R_2

STORE id_1, R_1

generate a target in form of 3-Address code.

(6)

③ $a = (b+c) * (b+c) * 2$

Construct all phases

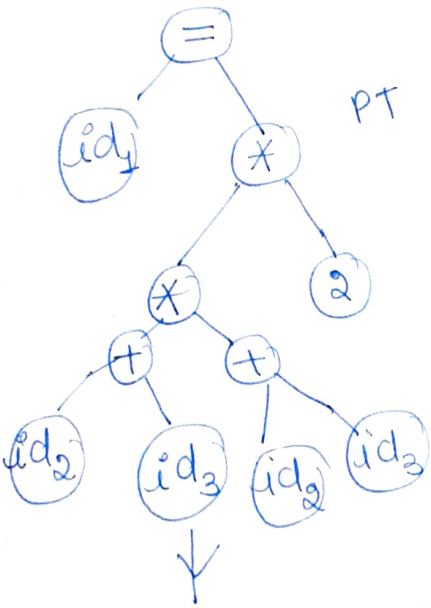
$a = (b+c) * (b+c) * 2$

↓ Lexical Analysis

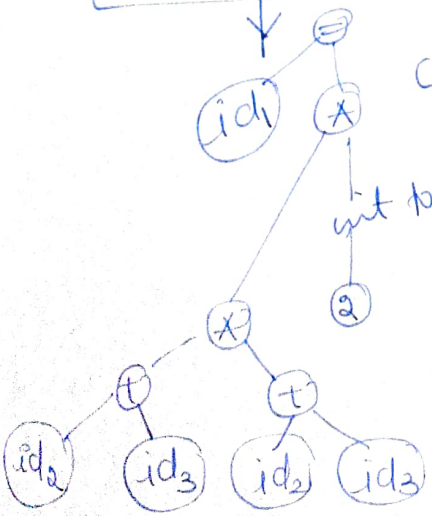
$id_1 = (id_2 + id_3) * (id_2 + id_3) * 2$

↳ Tokens

↓ Syntax Analysis



↓ Semantic Analysis



↓ Intermediate Code Generation

$t_1 = int\ to\ Real\ (2)$ — IC

$t_2 = id_2$ — 3-address code

$t_3 = id_3$

$t_4 = t_2 + t_3$

$t_5 = t_4 * t_4$

$t_6 = t_5 * t_1$

$id_1 = t_6$

↓ ~~Code Generation~~

↓ Code optimization

$t_1 = id_2 + id_3$

$t_2 = t_1 * t_1$

$id_1 = t_2 * 2.0$

optimized code in form of 3-address code

↓ Code Generation

```
LOAD  R1, id2
LOAD  R2, id3
ADD   R1, R2
MUL   R1, R2
MUL   R1, 2.0
STORE id1, R1
```

Relatively Target code in form of ALL.

Some other Question

① $i = i * 50 + j + 3$

② $a = (a+b) * 50 + j + z$

③ $p = i * r * t$