14/12/2023.

Sliptest - 1.

Part - A.

6. Identiby the number of tokens.

1) a = b ; → 4 tokens → (variable : a, b operator
2) a = = b ; → 5 tokens → variable : a, b
   operator : =, =
3) a + b ; → 4 tokens → variable : a, b
   operator : +
4) a ++ ; → 4 tokens → variable : a
   operator : ++
5) a < b ; → 4 tokens → variable : a, b
   operator : <.
6) a <= b ; → 5 tokens. → variable : a, b
   Operator : <, =

7. Consider the following program stateme

```
main ()
{
   int a, b;
   a = 5+8 + ;
   /* b = 5 */
}
```

How many number of tokens are P

8. What are reasons behind separating lexical analysis and parsing?

* Simpler design is the most important consideration

* Compiler efficiency is improved.

5. What advantages are there to divide a single pass into front end and back end in the phases of compiler?

* By keeping the same front end & attaching different back ends, one can produce a compiler for same source language on different machines.

* By keeping different front ends and same backend, one can compile several different languages on the same machine.

Part - A

1. Ans: c) only syntactical Error.
because fro is token as identifier in lexical analyzer. Where fro is not a keyword. So that it is only Syn-tactical Error.

2) Find the type of error produced by the following c code:

Ans: type error.

```
main()
{
   in /* common t x;
   floa /* comment */ t cse;
}.
```

3) In a compiler, keywords of a language are recognized during lexical analysis phase.

4) what are the advantages of (a) compiler over interpreter (b) an interpreter over a compiler?

a) the compiler compiles the whole program and produces the output accordingly the program. that are complied into the native machine code tend to be faster than interpretor code.

b) Interpretor languages tend to be more flexible and offer features like dynamic typing and smaller program size. Also because interpretor execute the source program code themselves, the code itself is platform independent.

9. To recover the errors in panic mode:-

1. Inserting a missing character
2. Deleting an extra character
3. Interchanging two adjacent characters
4. Replacing an incorrect character by a correct character.

28/35

7. Consider the following program statement

```
main()
{
    int a,b;
    a=5+8+;
    /* & b=5.*/
}
```

How many numbers of tokens are present in the above code,

main	<id,1>
(	<(>
)	<)>
{	<{>
int	<id,2>
a	<id,3>
,	<,>
b	<id,4>
;	<;>
a	<id,5>
=	<=>
5	<5>
+	<+>
8	<8>
+	<+>
;	<;>
}	<}>

there are 17 number of token.

22/25

8

Part - B

10. How the instruction x = y + z * 50 is passed to the compiler and how target code is generated from the given instruction. Explain the above with diagram.

Lexical analysis:

Source program → lexical analyzer.

x = y + z * 50 → lexical analyzer

↓

Output tokens

↓

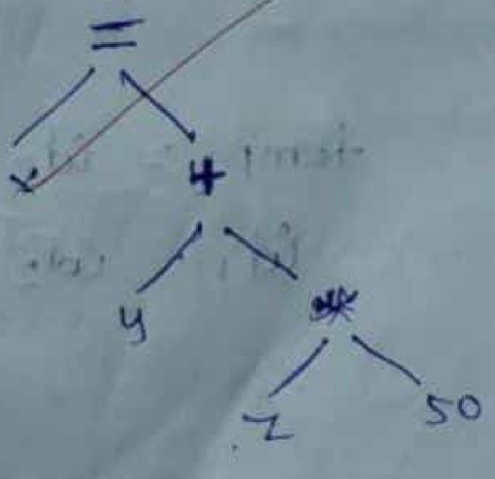
x , = , y , + , z , * , 50

Syntax analysis:

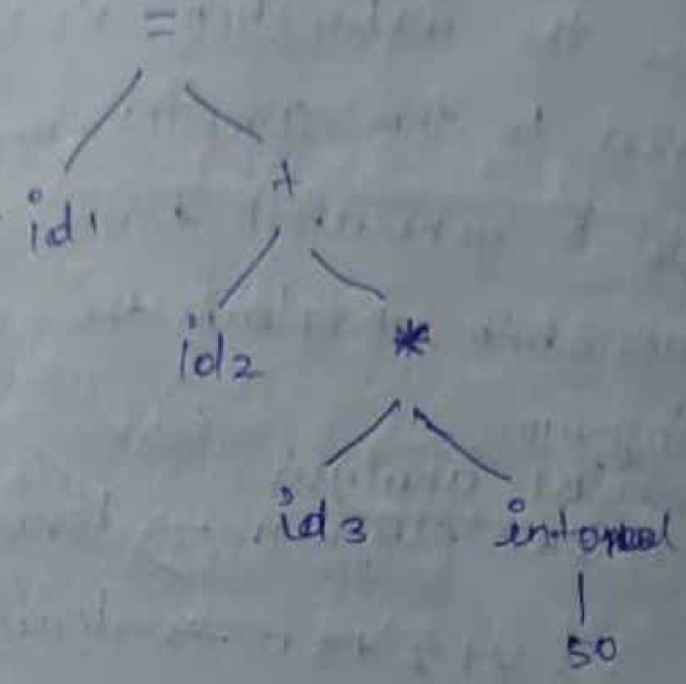
tokens → syntax analysis

↓

Syntax tree / parse tree.



Semantic analyzer:



Intermediate code generator:

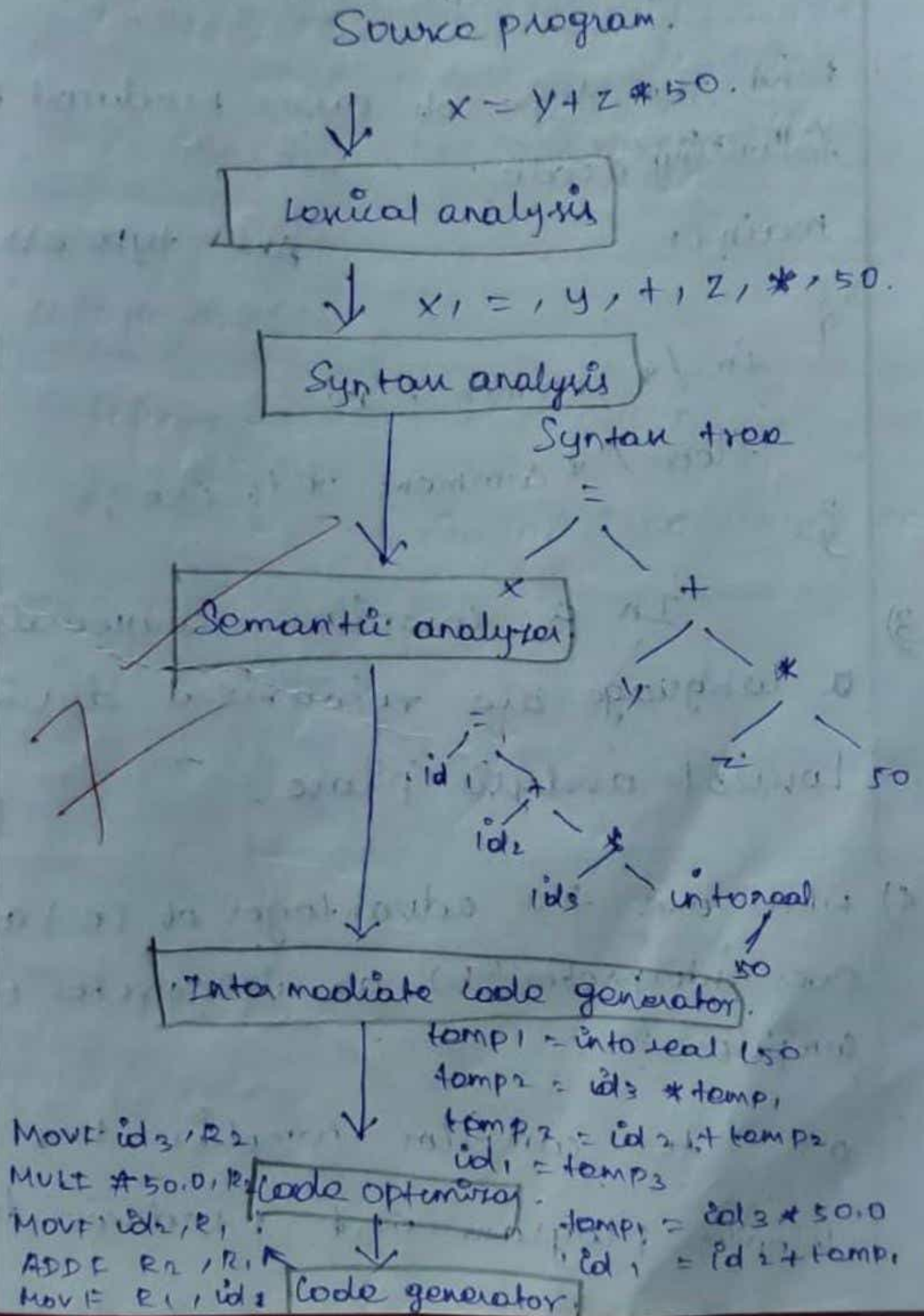
```
temp1 = intoreal (50)
temp2 = id3 * temp1
temp2 = id2 + temp2
id1 = temp3.
```

Code optimizer:

```
temp1 = id3 * 50.0
id1 = id2 + temp1
```

Code generator:

```
MOVF id3, R2
MULF #50.0, R2,
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1.
```



```
temp1 = inttoreal(50)
temp2 = id3 * temp1
temp3 = id2 + temp2
id1 = temp3
```

```
temp1 = id3 * 50.0
id1 = id2 + temp1
```

```
MOVF id3, R2
MULF #50.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
```