Slipter - 1

Part - A

Identify 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 : <, =

Consider the following program statement

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

How many number of tokens are P

What are reasons behind separating lexical analysis and parsing?

* Simpler design is the most important consideration
* Compiler efficiency is improved

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.

Ans: c) only syntactical error.

because ifre is taken as identifier in lexical analyzer. where ifre is not a keyword. so that it is only Syntactical Error.

---

find the type of error produced by the following c code:

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

Ans: type error

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

---

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 compiled into the native machine code tend to be faster than interpreter code

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

---

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.

Consider the following program segment

```
{
    int a, b;
    a = 5 + 8 * 1;
    /* & b = 5. */
}
```

How many number 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 >
*	< * >
1	< 1 >
;	< ; >
}	< } >

There are 17 number of tokens

Part - B :

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

↓

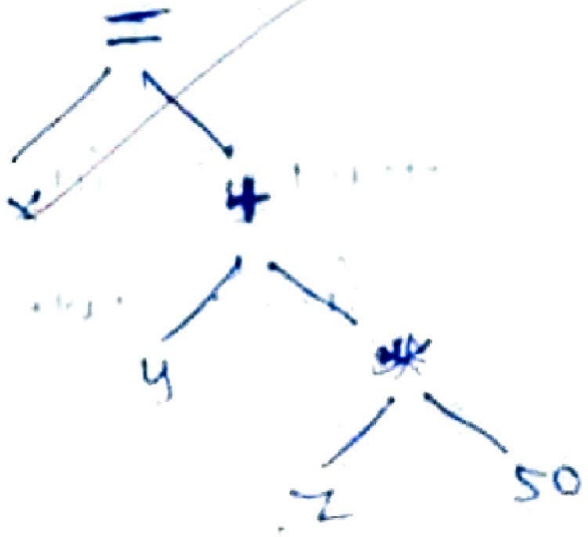
x , = , y , + , z , * , 50

Syntax analysis :

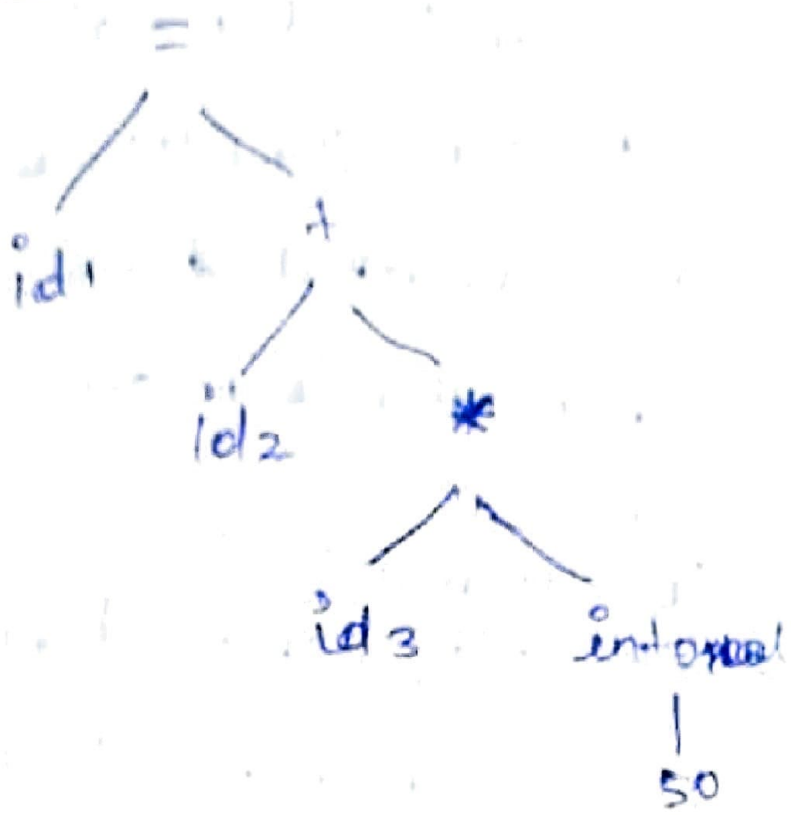
tokens → syntax analysis

↓

Syntax tree / parse tree.



Semantic analyzer:



Intermediate Code generator:

$temp_1 = inttoreal\ (50)$

$temp_2 = id_3 * temp_1$

$temp_2 = id_2 + temp_2$

$id_1 = temp_3$

Code optimizer:

$temp_1 = id_3 * 50.0$

$id_1 = id_2 + temp_1$

Code generator:

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