Language Description

- This language is called LAN. It should be capable of simple decimal algebra operations on integers.

- Example:

```
int a = 6;
int b = 10;
a += b;
int c;
c = a * b;
print(c); // 160
```

Language Description

- LAN also includes **if ... else** statements and **loops**
- Another example:

```
int a = 6;
while(a < 10) {
    a += 2;
}
if(a % 2 == 0) {
    print(1);
} else {
    print(0)
}
```

Lexical Tokens

- The concrete syntax of LAN is based on ASCII character encoding.

<identifier>	::=	[A-Za-z_][A-Za-z0-9_]*
<number>	::=	0 | [1-9][0-9]*
<unop>	::=	-
<binop>	::=	+ | - | * | / | %
<asnop>	::=	= | += | -= | *= | /

Precedence

- The precedence of unary and binary operators

Operator	Associates	Class	Meaning
-	Right	Unary	Negation
* / %	Left	Binary	Multiplication, division, modulo
+ -	Left	Binary	Addition, subtraction
= += -= *= /=	Right	Binary	Assignment

Reserved Keywords

- The following are reserved keywords or lexical tokens and cannot appear as a valid token in any place not explicitly mentioned as a terminal in the grammar.
  - if, else, true, false
  - while
  - int
  - print

Whitespace and Token Delimiting

- In LAN, whitespace is either a space or a carriage return (\r) in ASCII encoding.

- Whitespace is ignored, except that it terminates tokens. For example, += is one token, while + = is two tokens.

Language Syntax (1)

- The compiler translates source programs written in LAN. The syntax of LAN is defined by the context-free grammar shown here

- *Non-terminals are in <brackets>, terminals are in **bold**.

<block>	::=	**{** <statements> **}**
<statements>	::=	<block> | <statements> | <statement>
<statement>	::=	<declaration> **;** | <assignexpression> **;**
<declaration>	::=	**int identifier** | **int identifier =** <expression>
<expression>	::=	**identifier** | **number** | <expression> <binop> <expression> | <unop> <expression>
<assignexpression>	::=	**identifier** <asnop> <expression>

Language Syntax (2)

- The compiler translates source programs written in LAN. The syntax of LAN is defined by the context-free grammar shown here

- *Non-terminals are in <brackets>, terminals are in **bold**.

<conditionblock>	::=	**if (** <condition> **)** <block> | **if (** <condition> **)** <block> **else** <block>
<condition>	::=	**true** | **false** | <comparison>
<comparison>	::=	<expression> <cmpop> <expression>
<cmpop>	::=	**<** | **>** | **<=** | **>=** | **==** | **!=**
<loopblock>	::=	**while (** <condition> **)** <block>

Language Semantics

- Declaration
  - Though declarations are a bit redundant in a language with only one type, LAN requires every variable to be declared (with the correct type, in this case **int**) before being used.
- Operational semantics
  - The division i/k returns the truncated quotient of the division of i by k, dropping any fractional part. This means it always rounds towards zero.
  - The modulus i % k returns the remainder of the division of i by k.
  - Division i/k and modulus i % k are required to raise a divide exception if k = 0.