

Write a program to generate 3-address code for programs defined by the grammar below. +, -, \*, /, =, [], if, ifFalse, goto, !, <, >, <=, >=, ==, !=, param, call and ret are the allowed operators. Please submit ONLY the source files.

The interpretation of the *for* statement is as follows:

*for id = e1, e2 block* is equivalent to

*for (id = e1; id < e2; id++) block*

```

<program> -> class Program { <field_decl>* <method_decl>* }
<field_decl> -> <type> (<id> | <id> [ <int_literal> ] ) ( , <id> | <id> [ <int_literal> ] )* ;
<field_decl> -> <type> <id> = <literal> ;
<method_decl> -> ( <type> | void ) <id> ( ( <type> <id> ) ( , <type> <id> )* )? <block>
<block> -> { <var_decl>* <statement>* }
<var_decl> -> <type> <id> ( , <id> )* ;
<type> -> int | boolean
<statement> -> <location> <assign_op> <expr> ;
<statement> -> <method_call> ;
<statement> -> if ( <expr> ) <block> ( else <block> )?
<statement> -> for <id> = <expr> , <expr> <block>
<statement> -> return ( <expr> )? ;
<statement> -> break ;
<statement> -> continue ;
<statement> -> <block>
<assign_op> -> =
<assign_op> -> +=
<assign_op> -> -=
<method_call> -> <method_name> ( ( <expr> ( , <expr> )* )? )
<method_call> -> callout ( <string_literal> ( , <callout_arg> )* )
<method_name> -> <id>
<location> -> <id>
<location> -> <id> [ <expr> ]
<expr> -> <location>
<expr> -> <method_call>
<expr> -> <literal>
<expr> -> <expr> <bin_op> <expr>
<expr> -> - <expr>
<expr> -> ! <expr>
<expr> -> ( <expr> )
<callout_arg> -> <expr> | <string_literal>
<bin_op> -> <arith_op> | <rel_op> | <eq_op> | <cond_op>
<arith_op> -> + | - | * | / | %
<rel_op> -> < | > | <= | >=
<eq_op> -> == | !=

```

<cond\_op> -> **&&** | **||**  
<literal> -> <int\_literal> | <char\_literal> | <bool\_literal>  
<id> -> <alpha> <alpha\_num>\*  
<alpha> -> [**a-zA-Z**]  
<alpha\_num> -> <alpha> | <digit>  
<digit> -> [**0-9**]  
<hex\_digit> -> <digit> | [**a-fA-F**]  
<int\_literal> -> <decimal\_literal> | <hex\_literal>  
<decimal\_literal> -> <digit> <digit>\*  
<hex\_literal> -> **0x** <hex\_digit> <hex\_digit>\*  
<bool\_literal> -> **true** | **false**  
<char\_literal> -> '**<char>**'  
<string\_literal> -> "**<char>\***"