My own Python!

Your task is to design a program interpreter / error checker, based on object-oriented constructs taught till now. A code segment will be given as input. The interpreter must be able to perform a step-wise execution of the program while checking for any possible errors and reporting the same along with the corresponding line numbers (basically syntax and logical error checking). Assume that the codes given will have only two types of statements (assignment and print) and two types of operations (+ and -).

Format for valid statements

Please refer the following formats for valid statements. The language is case-sensitive.

- **Assignment:**
  <variable> = <expression>
- **Expression:**
  <variable> | <constant> | <expression> + <expression> | <expression> - <expression>
- **Variable:**
  <a-zA-Z_> <a-zA-Z0-9 >*
  (except for keyword `print`. Note the underscore symbol.)
- **Constant:**
  <0-9>+
  (Zero is a valid constant. There would be no testcase with leading zeros such as 002, or negative constants.)
- **Print:**
  print <varconstlist>
- **Variable list:**
  <varlist>: (<variable> | <constant>)+

Input Format

Assume that code statement/ line numbers start from 1 (N is immaterial here).
$< Code\_Statement\_1 >$
$\vdots$
$< Code\_Statement\_i >$
$\vdots$
$< Code\_Statement\_N >$
where $< Code\_Statement\_i >$ could be either an assignment statement or a print statement. All the tokens are space separated. i.e. there won't be any input like 'x+ 1' or 'x+1'

Constraints

Your code must have the following classes:

```
class Token {
      private:
         string name;
         int type; //0 for variable, 1 for constant, 2 for operator
         int value;
         //<type> variable; //any other you require
      public:
         bool is_initialized(); // for variable
         bool is_valid(); // for operator
         void operator +/-/= (Token const &b){
         //...your code...
         }
         //other_method();
   };
```

```
class Expression {
      private:
         bool is_valid; // for error checking
         bool is_lhs; // to check if there is an expression in LHS of assignment operator
         vector<Token> exp_tok;
      public:
       // Your methods go here
   };
```

You are free to add more data members, methods and classes along with aforementioned ones.

Output Format

For $< Code\_Statement\_i >$, the output should be:

- Print the appropriate output in case of a valid print statement.
- Valid print statement will have only variables in it (space separated variables, if more than 1 variable).
- In case of an invalid statement (print/ assignment), the output should be:
  line i: < appropriate ERROR MESSAGE as described below >
  Note: There is a single space after ':' and the ERROR message.
- For other valid assignment statements, simply perform the operation, no need for output.

Error Messages

These are the following types of ERROR MESSAGES to be reported

- If there exists a variable in the RHS of the assignment that is not initialized yet, then ERROR MESSAGE is: `<variable> is uninitialized`
- In case of multiple uninitialized variables on the RHS, the order of printing should be from left to right.
- If the LHS of an assignment statement is not a single variable but a constant or an expression, then ERROR MESSAGE is: `invalid l-value`
- If there is an expression in the print, then ERROR MESSAGE is: `expression not allowed for print`
- For any other non-assignment statement (like expression) or print with no arguments then the ERROR MESSAGE is: `invalid command`
- There could be multiple errors in a statement, but if present, they all would be of the same type..

Sample Input 0

```
w = 11
x = 0
y = 9
print x
print y
z = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1
print w x y z
w = 1 + 2 + 3 + 4 + 5 + 6
print z x w y
```

Sample Output 0

```
0
9
11 0 9 10
10 0 21 9
```

Sample Input 1

```
x = 1
y = 2
print x
```

```
print y
z = x + y
print z
w = z - 1
print w
v = x + y + z + w
print v
print v w x y z
```

Sample Output 1

```
1
2
3
2
8
8 2 1 2 3
```

Sample Input 2

```
qa5l = 8
qa5l = s9hi
qa5l = l3_l
print
wywo = 7
print wywo
print wywo
lrw2 = 5
print lrw2
print lrw2
kchc = 8 - 9
print wywo
_s0k = 2 - 2 + qa5l
gu12 = 9 - 7 + 3 - 8 + 6 - 8
print lrw2 gu12
kchc = b5q2
ysh1 = 8 - lrw2 + 8 + 1 - 5 - _s0k - 5 + 8
gu12 - 8 + 8 - 9
cuko = gu12
vuf4 = 4 + 3 - 8 - 2 - 4 + 6 - 1 - 7 + 2 - 7 + 6
nqen = lrw2
print qa5l + 7
mpzv = cuko + qa5l + 9 + kchc + 0 + 1 - 6 + _s0k - 8 + lrw2 + 4
cuko = vd4y
mpzv = dbl0
ll6r = gu12 + lrw2 + 1 - ysh1 + 9 + 8 - 6 + 6 - 3 + 0 - 0 + 0 - qa5l + 4
qdg_ = ysh1 + kchc - 1 - ysh1 - ysh1 + ysh1 - ysh1 - wywo - 2 + 0 - 2 + _s0k + _s0k - 6 + vuf4 - ysh1 - gu12
ll6r = frt9 + h0u3 + lb9b
kchc - cuko = vuf4
print wywo lrw2 qa5l gu12 mpzv nqen _s0k vuf4 ll6r
print 7 - lrw2 + 9 - ll6r + cuko + mpzv + wywo
print gu12 vuf4 nqen qdg_
print wywo _s0k qdg_ mpzv kchc ll6r gu12 ysh1 qa5l
print gu12 wywo qdg_ ysh1 mpzv kchc
print 0 - ll6r - wywo - ysh1 - gu12 + cuko - kchc + qa5l - 3 - 6 + gu12
e_uu = cuko
u8gm = wywo - qa5l + 8 + cuko - vuf4 + 0 + ll6r - 5
qa5l = t879
ysh1 = bdn3 + bghr - sz6d - _wlp
qa49 = cuko - nqen - gu12 + ll6r - _s0k
ppeh = 0 - lrw2 + 7 + wywo - _s0k - 1 + 4 + 4 - 5 + qdg_ - 9
n1yx = nqen + 4 + vuf4 + _s0k + 7 + cuko + ll6r - u8gm + 3 + 8 + 6 - 6 + ysh1 - lrw2 - 1 + 0 - mpzv - vuf4 - _s0k - qdg_ + wywo
+ vuf4
_fom = 6 - lrw2 - gu12 + cuko - qa49 - vuf4 + 0 + 5
print ysh1 kchc e_uu gu12 u8gm ll6r ppeh cuko _fom n1yx qa49 lrw2 nqen mpzv wywo
print gu12 qa49 mpzv e_uu ppeh vuf4 ysh1 qdg_ ll6r lrw2 qa5l wywo _fom kchc nqen u8gm _s0k
j9g4 = mpzv - u8gm - n1yx - lrw2 - 1 + kchc + _s0k - ppeh + 3 - qa49 - n1yx
ll6r = ygt3 - a44o + dnx9 - pov8 - p4dm - uzhy + a68m + sa5w - tivm
print wywo j9g4 ll6r kchc qa49 u8gm vuf4 _s0k ysh1 _fom mpzv
my4_ = ysh1 - lrw2 + 2 - kchc - ysh1 - qa5l + 9 + u8gm - 9 + 9 - 4 + _fom + 0 + ll6r - 8 - nqen + nqen + 1 + qa5l + wywo + 7 -
qdg_
print kchc _s0k _fom qdg_ ll6r
```

Sample Output 2

```
line 2: s9hi is uninitialized
```

line 3: l3_l is uninitialized
line 4: invalid command
7
7
5
5
7
5 -5
line 16: b5q2 is uninitialized
line 18: invalid command
line 22: expression not allowed for print
line 24: vd4y is uninitialized
line 25: dbl0 is uninitialized
line 28: frt9 is uninitialized
line 28: h0u3 is uninitialized
line 28: lb9b is uninitialized
line 29: invalid l-value
7 5 8 -5 15 5 8 -8 9
line 31: expression not allowed for print
-5 -8 5 -10
7 8 -10 15 -1 9 -5 2 8
-5 7 -10 2 15 -1
line 35: expression not allowed for print
line 38: t879 is uninitialized
line 39: bdn3 is uninitialized
line 39: bghr is uninitialized
line 39: sz6d is uninitialized
line 39: _wlp is uninitialized
2 -1 -5 -5 14 9 -16 -5 18 7 -4 5 5 15 7
-5 -4 15 -5 -16 -8 2 -10 9 5 8 7 18 -1 5 14 8
line 47: ygt3 is uninitialized
line 47: a44o is uninitialized
line 47: dnx9 is uninitialized
line 47: pov8 is uninitialized
line 47: p4dm is uninitialized
line 47: uzhy is uninitialized
line 47: a68m is uninitialized
line 47: sa5w is uninitialized
line 47: tivm is uninitialized
7 11 9 -1 -4 14 -8 8 2 18 15
-1 8 18 -10 9