189 Pascal Program Lengths

Your local computer user's group publishes a quarterly newsletter, and in each issue there is a small Turbo Pascal programming problem to be solved by the membership. Members submit their solutions to the problem to the newsletter editor, and the member submitting the shortest solution to the problem receives a prize.

The length of a program is measured in units. The unit count is determined by counting all occurrences of reserved words, identifiers, constants, left parentheses, left brackets, and the following operators: `+`, `-`, `*`, `/`, `=`, `<`, `>`, `<=`, `>=`, `<>`, `@`, `^`, and `:=`. Comments are ignored, as are all other symbols not falling into one of the categories mentioned above. The program with the lowest unit count is declared the winner. Two or more programs with equal unit counts split the prize for the quarter.

In an effort to speed the judging of the contest, your team has been asked to write a program that will determine the length of a series of Pascal programs and print the number of units in each.

Input and Output

Input to your program will be a series of Turbo Pascal programs. Each program will be terminated by a line containing tilde characters in the first two columns, followed by the name of the submitting member. Each of these programs will be syntactically correct and use the standard symbols for comments (braces) and subscripts (square brackets).

For each program, you are print a separate line containing the name of the submitting member and the unit count of the program. Use a format identical to that of the sample below.

Sample input

```
PROGRAM SAMPLEINPUT;

VAR
  TEMP : RECORD
    FIRST, SECOND : REAL;
    END;

BEGIN {Ignore this }
TEMP.FIRST := 5.0E-2;
READLN (TEMP.SECOND);
WRITELN ('THE ANSWER IS', TEMP.FIRST * TEMP.SECOND : 7 : 3)
END.
~~A. N. Onymous
```

Sample output

```
Program by A. N. Onymous contains 29 units.
```

Note: Here are some additional notes on Turbo Pascal for those not familiar with the language:

- Identifiers start with an underscore (_) or a letter (upper or lower case) which is followed by zero or more characters that are underscores, letters or digits.

- The delimiter for the beginning and ending of a string constant is the single forward quote (´). Each string is entirely on a single source line (that is a string constant cannot begin on one line and continue on the next). If ´´ appears within a string then it represents a single ´ character that is part of the string. A string constant consisting of a single ´ character is, therefore, represented by ´´´´ in a Turbo Pascal program. The empty string is allowed.
- The most general form of a numeric constant is illustrated by the constant `10.56E-15`. The `10` is the integral part (1 or more digits) and is always present. The `.56` is the decimal part and is optional. The `E-15` is the exponent and it is also optional. It begins with an upper or lower case `E`, which is followed by a sign (`+` or `-`). The sign is optional.
- Turbo Pascal supports hexadecimal integer constants which consist of a $ followed by one or more hex digits ('0' to '9', 'a' to 'f', 'A' to 'F'). For example, $a9F is a legal integer constant in Turbo Pascal.
- The only comment delimiters that you should recognise are {}, and not `(**)`. Comments do not nest.
- '+' and '−' should be considered as operators wherever possible. For example in `x := -3` the '−' and the '3' are separate tokens.
- Subranges of ordinal types can be expressed as `lower..upper`. For example, `1..10` is a subrange involving the integers from 1 to 10.
- All tokens not mentioned anywhere above consist of a single character.