4444 Swamp County Utility Generator

To celebrate the 45th anniversary of Swamp County's first computer, an IBM 1401, class assignments this term will have an "oldies" theme. The project for your team is to implement a SCUG interpreter, emulating the first utility written by the Swamp County programmers. Back then the high-level languages for the 1401, FARGO and RPG, required 4000 characters of core memory but Swamp County could only afford the entry level model with 1400 characters. All the larger production programs had to be shipped off to a service bureau and the resulting executable decks shipped back.

SCUG was a primitive programming language used to classify, compare, and rearrange data on punch cards. See the SCUG manual that follows for the details of the language.

Input

Input to your interpreter will begin with the program to be executed. The program will consist of 80-character lines each terminated by end-of-line. The program will end with an 80-character line starting with '#'.

The input for the program to be executed follows and will consist of 80-character lines each terminated by end-of-line. These lines will be terminated by end-of-file.

For the purposes of this project, the characters are normal ASCII, not the more historically correct 1401 alphameric codes. Card column 1 is the first character on a line.

The judged and provided sample input are correct and should generate no errors.

Output

The results from the card punch, if any. Each card image is an 80-character line terminated by end-of-line. Each line must contain 80 characters before the end-of-line.

Explanation:

```
                  SWAMP COUNTY UTILITY GENERATOR

A SCUG program reads and outputs 80-character cards.  Card columns
are numbered from the left from 1 to 80.

In this write up, 01-80 means '01' or '02' or ... '80'. F0-F9 means
'F0' or 'F1' or ... 'F9'. Similarly for V0-V9 etc.

A SCUG program consists of three types of definition statements:
variable, card type, and card field; comment statements; and
executable action statements. The definition statements (variable,
card type, card field) must appear before any executable statements.
There can be only one card field definition in a program. Comments
may appear anywhere in the program card deck.

SCUG programs run in what is called the "program cycle".  The program
executes within an implied loop for each input card:  the card type
definitions and executable statements are applied to each input card
in the order they appear in the program.
```

```
If there is an error, processing will halt. The operator will indicate
the last card read, which caused the error.

SCUG has an 80 character card input buffer CB, an 80 character card
punch output buffer PB, up to ten variables, V0..V9, and up to ten
card types C0..C9 which are set by the card in the input buffer.
There can be up to ten card fields defined in CB: F0..F9.

PB is set to all blanks each time a new card is read into CB, but
NOT after each time PB is punched out.

Variables can be up to 80 characters long.

The type of a statement is determined by the character in the column 1:
  '@' Comment
  'V' Variable Definition
  'C' Card Type Definition
  'F' Card Field Definition
  'A' Action Executable

Card columns 77 through 80 are reserved for optional sequence
numbers.

Comment
  col 1      '@'
  col 2..76  any characters.

Variable Definition
  col 1       'V'
  col 2       0-9 the variable to be defined. Each variable may only be
              defined once in a program with its initial value.
  col 3-4     01-80 length of the variable, a leading zero is required
  col 5-76    initial value, left adjusted
  If the variable length is longer than 72 characters, it is filled with
  blanks on the right. If the variable length is shorter than 72 characters
  the initial value is truncated on the right.

Card Type Definition
  col 1       'C'
  col 2       0-9 the card type to be defined.  There may be any number
              of definitions of a given card type but during execution the
              first type that satisfies the tests that follow on the
              definition card will be assigned.
  col 3-74    up to 18 character tests. If all tests are true, the card is
              of this type. If there are several variants of a card type,
              include a definition for each variant.
    The first character test:
      col 3       ' ' or 'N'. If 'N' the test is true if the character in
                  the specified card column does NOT match.
      col 4-5     01-80 the card column to be tested. a leading '0' is
                  required.
      col 6      the character to be matched
      If all 4 columns of a test are blank, the test is true.
    The rest of the tests start in columns: 7, 11, 15, 19, 23, 27, 31, 35,
    39, 43, 47, 51, 55, 59, 63, 67, 71.

Card Field Definition
  col 1       'F'
  col 2-41    field definitions, 0 through 9
    The first definition (for F0):
      col 2-3     start column in CB for F0
      col 4-5     end column in CB for F0
    The rest of the definitions start in columns: 6, 10, 14, 18, 22, 26,
    30, 34, 38.
  If a field is not to be defined, leave its columns blank. There
  can be only one card field definition in a program. It is an error
  if the end column is less than the start column.

Action Executable
  col 1       'A'
  col 2-3
    'PB' destination is PB
     V0..V9 destination is the variable
  col 4       ' ' or 'P'. If 'P' punch a card from PB if the conditions
              are true and after the operations are executed.
  col 5-13    up to 3 condition tests.  The operations of the statement
              are only executed if all included conditions are true.
    The first condition test:
      col 5       ' ' or 'N'. If 'N' invert the condition.
      col 6-7     the condition:
        C0..C9 true if the card in CB has been assigned that card type
               definition
        B0..B9 break--true if the next card to be read is the same type as
               the card in CB but the corresponding field, F0..F9
               differs; or the next card to be read has a different
               type than the card in CB; or the card in CB is the last
               card.
         '  ' true
    The rest of the conditions start in columns: 8, 11.
  col 14-76   up to 9 operations. They are executed from left to right.
    The first operation:
      col 14      op-code:
        ' ' do nothing, the factor and destination columns are ignored
        'L' move the factor to the result.  The factor will be truncated
            or blank filled on the right if necessary.
           '1234' -> '123' or '1234' -> '1234  '
        'R' move the factor to the result.  The factor will be truncated
            or blank filled on the left if necessary.
           '1234' -> '234' or '1234' -> '  1234'
        'B' set result to all blanks.
       col 15-16   factor
         F0..F9 a field in CB
         V0..V9 a variable
         '  ' if op-code does not require a factor
       col 17-18   01-80 starting column in destination to store result. A
                   leading '0' is required.
       col 19-20   01-80 ending column in destination to store result. A
                    leading '0' is required.
      It is an error if the starting or ending column is greater
      than the length of the destination or the end column is less than
      the start column.
    The rest of the operations start in columns: 21, 28, 35, 42, 49, 56,
    63, 70.
```

Sample Input

```
@ duplicate the deck
F0180
APBP         LF00180
#
this is the first card                                                  0010
this is the second card                                                 0020
this is the third card                                                  0030
```

Sample Output

```
this is the first card                                                  0010
this is the second card                                                 0020
this is the third card                                                  0030
```