ACM-style Programming Contest
Team Selection Trials
Contest Two — Oct. 5, 1996

Rules:

1. All questions requre you to read the test data from standard input and write results to standard output.
   - No whitespace should appear at the end of a line, and all lines should be terminated with a new-line.
   - Tabs should never be used.
   - Output must correspond *exactly* to the provided sample output, including spelling and spacing. Multiple spaces will not be used in any of the judges output, except where expressly stated.
2. All programs will be re-compiled prior to testing with the judges' data. For this reason, non-standard libraries should not be used in your solutions.
3. Programming style is not considered in this contest. You are free to code in whatever style you prefer.
4. All communication with the judges will be handled by the submit command. If you have need of books, ask the helpers in the room.
5. Judges' decisions are to be considered final. No cheating will be tolerated.
6. There are five questions to be completed in three hours.

Question 1 — The Game of 31

The game of 31 was a favourite of con artists who rode the railroads in days of yore. The game is played with a deck of 24 cards: four labelled each of 1, 2, 3, 4, 5, 6. The cards in the deck are visible to both players, who alternately withdraw one card from the deck and place it on a pile. The object of the game is to be the last player to lay a card such that the sum of the cards in the pile does not exceed 31. Your task is to determine the eventual winner of a partially played game, assuming each player plays the remainder of the game using a perfect strategy.

For example, in the following game player B wins:

```
Player A plays 3
Player B plays 5
Player A plays 6
Player B plays 6
Player A plays 5
Player B plays 6
```

Input:

The input will consist of several lines; each line consists of a sequence of zero or more digits representing a partially completed game. The first digit is player A's move; the second player B's move; and so on. You are to complete the game using a perfect strategy for both players and to determine who wins.

Output:

For each game, print a line consisting of the input, followed by a space, followed by A or B to indicate the eventual winner of the game.

Example Input:

```
356656
35665
3566
111126666
552525
```

Example Output:

```
356656 B
35665 B
3566 A
111126666 A
552525 A
```

Question 2 — Fibonacci Numbers

A Fibonacci sequence is calculated by adding the previous two members of the sequence, with the first two members being both 1.

$$f(1) = 1, f(2) = 1, f(n > 2) = f(n-1) + f(n-2)$$

Your task is to take a number as input, and print that fibonacci number.

Sample Input:

```
100
```

Sample Output:

```
354224848179261915075
```

Note:

No generated fibonacci number in excess of 1000 digits will be in the test data, i.e. $f(20) = 6765$ has 4 digits.

Question 3 — Ransom Note

Gilbert Bates, the magnate of aluminum siding, doors, and windows, has been kidnapped. You are to help the perpetrators produce a ransom note. Your raw materials are the text of a newspaper and the text of the ransom note. The ransom note is to be produced by clipping letters or strings of letters (and possibly spaces) from the newspaper and pasting them onto a blank sheet of paper to form the note. Your job is to determine the minumum number of pieces of paper that must be clipped and pasted to form the note. Between each pair of words in the note, either the clipping must contain a space or a boundary between clippings must occur (so that the blank background shows through).

Input:
Standard input consists of the text of the note followed by the text of the newspaper. The text of the note is a single line, less than 20KB in length, in lowercase with no punctuation. The text of the newspaper is in upper and lower case with punctuation and newlines mixed in. Case may be ignored (aS IN aNY stANDard RAnsoM nOTE) and punctuation and newlines should not be clipped. The kidnappers have acquired a large number of copies of the same newspaper, so that the same or overlapping text may be clipped as many times as necessary. Every letter of the alphabet occurs at least once in the newspaper. The newspaper is smaller than 100KB in length.

Output:
Print the number of clippings followed by the clippings, one per line, in the correct order to compose the note. The case of the newspaper text should be preserved.

Sample Input:

```
drop the price on new thermopanes now or else
Rain Users Guide
While "rain" was intended to be a general purpose tool, at the time of
writing the primary goal was to study one particular software system.
As a result, some steps that are only done once (such as extracting
information from the program under study) are done using cumbersome
ad-hoc techniques that require significant manual intervention.  While
"rain" can be used on arbitrary programs, more development work needs
to be done before this is a convenient process.
```

Sample Output:

```
19
d
ro
p
the pri
ce
on
ne
w
the
rm
op
an
es
n
o
W
or
el
se
```

Question 4 — Scheme Pretty-Printing

Scheme is an expression language. This means that everything that is entered to the Scheme interpreter/compiler is an expression. Expressions are separated by blank space (blank, tabs, new-lines). Expressions can be in several forms:

1. numbers - a sequence of digits, possibly preceded by a '`+`' or '`-`', possibly including a single '`.`' following at least one digit
2. strings - a sequence of characters (not including newlines) preceded and followed by ", with any included double-quote characters preceded by a '`\`', such as `"This␣'\"'␣is␣a␣double-quote"`
3. special constants - a '`#`' followed by any characters up to blank space
4. compound expressions - a (possibly empty) sequence of expressions surrounded by parentheses
5. identifiers - a sequence of non-blank characters not including the characters: `#`, `"`, `\`, `(`, and `)`

The Input:

A sequence of Scheme expressions.

The Output:

The same sequence of expressions, reformatted to make them more readable. The rules you must follow are:

1. All top level expressions will start with no leading blanks on a line.
2. A compound expression with the first sub-expression being the identifier `define` is a define-form. The second sub-expression will be an identifier and should go on the same line as the word `define`. If the third (and last) expression is compound it should start on the following line, indented 3 spaces. Otherwise, the whole define-form should be on a single line.
3. A compound expressions with the first sub-expression being the identifier `lambda` is a lambda-form. The second sub-expression will be an identifier or compound expression and should go on the same line as the word `lambda`. All subsequent expressions should start on a new line, indented by an additional 3 spaces
4. A compound expressions with the first sub-expression being the identifier `if` is an if-form. The second sub-expression will be an identifier or compound expression and should go on the same line as the word `if`. All subsequent expressions should start on a new line, indented by an additional 4 spaces
5. All other compound expressions are function applications. If any of the sub-expressions are compound, the first two sub-expressions will be on the same line and all subsequent sub-expressions will be on new lines, indented to align with the second sub-expression.
6. In all other cases, all blank space between elements of a compound expression will be replaced by a single space.

Sample Input:

```
(define abc+ (lambda (@1 $f) (if (if
$f a     b) (@1
 3 4) (b (d e) (f "g")))))
           (define
     a 42)
(+ a (- b c))
```

Sample Output:

```
(define abc+
   (lambda (@1 $f)
      (if (if $f
              a
              b)
          (@1 3 4)
          (bcdefg (d e)
                  (f "g")))))
(define a 42)
(+ a
   (- b c))
```

Question 5 — Substitution Cypher

Substitution cyphers are the simplest of cyphers where the letters of one alphabet are substituted for the letters of another alphabet. In one form or another, they've been in use for over 2000 years.

Input:

- a line containing the plaintext alphabet
- a line containing the substitution alphabet
- several lines of text

Output:

- a line containing the substitution alphabet
- a line containing the plaintext alphabet
- the converted lines of text

Please note: All lines will be at most 64 characters, plus a trailing end-of-line character. Pass through all characters not found in the plaintext alphabet.

Sample Input:

```
abcdefghijklmnopqrstuvwxyz
zyxwvutsrqponmlkjihgfedcba
Shar's Birthday:
The birthday is October 6th, but the party will be Saturday,
October 5.  It's my 24th birthday and the first one in some
years for which I've been employed.  Plus, I have new clothes.
So I have cause to celebrate.  More importantly, though,
we've cleaned the house!  The address is 506-D Albert Street.
Extra enticement for CS geeks:  there are several systems in
the house, and the party is conveniently scheduled for 3 hours
after the second CSC programming contest ends (not to mention,
within easy walking distance)!
```

Sample Output:

```
zyxwvutsrqponmlkjihgfedcba
abcdefghijklmnopqrstuvwxyz
Sszi'h Brigswzb:
Tsv yrigswzb rh Oxglyvi 6gs, yfg gsv kzigb droo yv Szgfiwzb,
Oxglyvi 5.  Ig'h nb 24gs yrigswzb zmw gsv urihg lmv rm hlnv
bvzih uli dsrxs I'ev yvvm vnkolbvw.  Pofh, I szev mvd xolgsvh.
Sl I szev xzfhv gl xvovyizgv.  Mliv rnkligzmgob, gslfts,
dv'ev xovzmvw gsv slfhv!  Tsv zwwivhh rh 506-D Aoyvig Sgivvg.
Ecgiz vmgrxvnvmg uli CS tvvph:  gsviv ziv hvevizo hbhgvnh rm
gsv slfhv, zmw gsv kzigb rh xlmevmrvmgob hxsvwfovw uli 3 slfih
zugvi gsv hvxlmw CSC kiltiznnrmt xlmgvhg vmwh (mlg gl nvmgrlm,
drgsrm vzhb dzoprmt wrhgzmxv)!
```