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 `)`

Input

A sequence of Scheme expressions.

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) (bcdefg (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))
```