2054 Dynamic Declaration Language (DDL)

DDL is a very simple programming language in which variables are dynamically declared at run time. All variables in DDL are of the signed integer type within the range 99999999. There are up to five types of statements in a DDL program (each statement is in a separate program line, and the first statement is in line 1):

1. `Dcl` $< id >$

   `Dcl` is a keyword specifying a declaration statement. id is a single (case-sensitive) letter designating a DDL variable. For example '`Dcl` x' when executed correctly, allocates memory for variable x, and sets its value to zero.

2. $< id >$ `=` $< ic >$

   This is an assignment statement, where id is a DDL variable, and ic is a literal integer constant in the range $(0 \ldots 9999)$. For example $x = 2000$ when executed correctly, changes value of x to 2000. Note that there may be one or more number of blank characters around '`=`', but there is no tab characters.

3. `Goto` $< label >$, or `Goto` $< id >$ $< label >$

   `Goto` is a keyword specifying an unconditional or conditional goto statement. label is a program line's number. For example '`Goto` 5' transfers the program execution flow to line 5 of the program, and '`Goto` x 5' when executed correctly, trasfers the flow to line 5 iff $x > 0$, and to the next line otherwise. The label is guaranteed to be in the range of program line numbers.

4. `Inc` $< id >$, or `Dec` $< id >$

   `Inc` and `Dec` are keywords specifying increment and decrement statements respectively. For example '`Inc` x' ('`Dec` y') when executed correctly adds (substracts) 1 to (from) the value of x (y).

5. `End`

   `End` is a keyword specifying the end statement, whose execution stops the program.

Note that the keywords of the DDL language are case-insensitive.

Error conditions:

When one of the following erroneous statements encounters during the program execution, an error message appears in a separate line of the output. Each error message is of the form $< label > < space > < error\ code >$. *label* is the line number for the erroneous statement, *space* is one blank character, and *error code* is a positive integer specified below.

1. `Dcl` x is erroneous if x has not been referenced (used in assignment, goto, increment or decrement) since the last time a '`Dcl` x' (declaring the same variable) statement has been executed, unless this is the first '`Dcl` x' statement being executed. In this erroneous condition, an error message indicating a repeated declaration is generated as $< label > 1$, where label is the program line number for the erroneous statement. Then the program flow transfers to the statement in the next program line, and any prior correctly executed declaration for x is valid.

2. Any other statement where a variable such as x is referenced (used in assignment, goto, increment or decrement) is erroneous if no '`Dcl` x' has been previously correctly executed. In this case, an error message indicating an undeclared reference is generated as $< label > 2$ and the program execution continues from the next line.

Input

First line of the input file contains a single integer N indicating the number of DDL programs to follow ($1 \leq N \leq 20$).

The first line of each test case contains a single integer indicating number of statements in that program which is in the range ($1 \ldots 100$). There are no blank lines between test cases. Statements of each DDL program come one after the other in separate lines without any blank lines in between. Statements are not explicitly labeled, but they are implicitly labeled by the number of their line beginning from 1 for the first statement in each program. There is no syntax error in programs and they are guaranteed to terminate, and no overflow or underflow errors will occur during execution. In each line of the program, tokens (e.g. '`GOTO`', '`=`', etc.) are separated by at least one blank character. Also there may be some blank characters in the beginning or at the end of each line.

Output

For each input DDL program, your output should start with the program number in the first line, followed by the error messages generated by the program in the order they are generated, each error message in one line. There should be no blank lines between error messages.

Sample Input

```
2
4
DCL X
INC X
DCL X
END
9
DCL X
INC X
GOTO X 5
DCL Y
Y = 100
DCL X
DCL X
Y = 50
END
```

Sample Output

```
1
2
5 2
7 1
8 2
```