While reviewing code recently being checked into the repository, Jim discovered that some employees now and then seemed to copy code right from the Internet into the company code base. This would be a potential disaster as the company then risks getting sued by copyright holders of the original code. The obvious solution, talking to the employees and kindly ask them not to submit any stolen code, seemed to solve the problem. Still, it was decided that a screening process should be introduced to detect newly stolen code.

The screening would work as follows: Every time new code was checked in, the full contents of the changed files where matched against a repository of known open source code. For each file the longest match, in number of consecutive lines, should be reported.

Comparison is done line by line. Empty lines, and lines only containing space, are ignored during comparison and not counted. Leading and trailing spaces should be ignored completely and consecutive space characters inside the lines are treated as one single space. The comparison is case-sensitive.

Input

The input file contains several test cases, each of them as described below.

Test data starts with the number $0 \leq N \leq 100$ of code fragments in the repository. Then follows, for each code fragment, one line containing the file name that the fragment was fetched from and the contents of the fragment on subsequent lines. File names will neither contain whitespace nor be guaranteed to be unique. The name is at most 254 characters long. Each fragment is terminated by `***END***` on a line by its own. This line is not considered being part of the fragment.

After the fragments in the repository have all been listed, comes the actual code snippet to find matches for. This snippet is also terminated by `***END***` on a line by its own.

Lines are guaranteed to be no longer than 254 characters. No code fragment will be longer than 10000 lines. Any code and file name lines will only contain the ASCII characters 32-126. The total size of the input file will not exceed 10^6 characters.

Output

For each test case, write to the output, on a line by itself, the number of matching consecutive lines (empty lines not counted) in a longest match from the repository, followed by a space-separated list of the file names of each fragments containing a match of this length, given in the order that the matching fragments were presented in the repository description. If no fragments match, write the number 0 on a line of its own.

Sample Input

```
2
HelloWorld.c
int Main() {
    printf("Hello %d\n",i);
}
***END***
Add.c
int Main() {
  for (int i=0; i<10; i++)
    sum += i;
  printf("SUM %d", sum);
}
***END***
int Main() {
  printf("Hello %d\n",i);
  printf("THE END\n");
}
***END***
2
HelloWorld1.bas
10 PRINT "******************"
20 PRINT "******************"
30 PRINT "--- HELLO WORLD ---"
40 PRINT "******************"
50 PRINT "******************"
***END***
HelloWorld2.bas
10  PRINT   "------------------"
20  PRINT   "******************"
30  PRINT   "--- HELLO WORLD ---"
40  PRINT   "******************"
50  PRINT   "------------------"
***END***
10 REM Hello ver 1.0 (c) Acme 2008
20 PRINT "******************"
30 PRINT "--- HELLO WORLD ---"
40 PRINT "******************"
50 END
***END***
```

Sample Output

```
2 HelloWorld.c
3 HelloWorld1.bas HelloWorld2.bas
```