In the game of Boggle, you are presented with an $N \times N$ table of letters. The object is to find words in the mix of letters. A word may be started anywhere in the table and is constructed by forming a chain of adjacent letters. Adjacent means diagonal, vertical or horizontal. A word cannot use any character from the table more than once.

Here is an example of a 4×4 table:

```
bile
tglp
aest
here
```

The following is a partial list of legal words that can be found using the above rules:

```
bill
gates
slept
here
```

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

In "Boggle Blitz" you will be given an integer number, N, on a single line. N is the number of characters on each side of the table, giving a total of N^2 characters. N can range from 1 to 20. N lines of N characters each will follow giving you the arrangement of the table.

Your task is to find all the legal words in the table. **Be sure that your program is efficient!**

In case you forgot to bring a dictionary, you are in luck because we are not using English words. Instead, we have redefined "word" to mean an increasing (by ASCII value) chain of characters from length 3 to length N^2. "`ABCDE`" is a legal five letter word. "`MICROSOFT`" is **not legal** because the sequence is not increasing. "`BILL`" is also illegal because `L` is not greater than `L`. "`BIL`", however, is legal.

Sample Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Your program should output the list of unique words sorted according to the following criteria:

1. Shorter words are before longer words
2. Words are sorted lexigraphically by ASCII value
3. Add no blank lines or spaces. If no legal words can be found, print nothing.

Sample Input

```
2

3
one
top
dog

4
abcd
bcda
cdab
dabc
```

Sample Output

```
dop
dot
eno
enp
ent
eop
eot
gop
got
nop
not
enop
enot

abc
abd
acd
bcd
abcd
```