Nowadays everyone is using compression methods to reduce the space occupied by data. In some cases this is done in such a way you hardly notice it, for example tapestreamers, modems and harddisk doubling programs. In other cases you have to do it yourself by using *pack*, *arj*, *zip*, *zoo* or *arc*.

Most compression-methods are very complicated, but for this problem only the following simplified method is used. An ASCII-character consists of eight bits, which allows the encoding of 256 different characters. It's possible to recode the characters with a new sequence of bits. These sequences may have a different length. When you choose to give the characters which are often used a shorter sequence of bits than the characters which are seldom used the total text size will be reduced.

Recoding characters gives one other problem, which occurs when you try to get the text back. In a standard ASCII text you know the first character begins at the 1st and ends at the 8th bit, the second begins at the 9th and ends at the 16th bit and so on. When using a variable length coding you don't know where a character begins. For example when an '`a`' is coded as '11' and an '`e`' as '1111', given the bit-sequence '111111' you don't know if it means '`ae`', '`ea`' or '`aaa`'.

The last problem is solved when the next principle is used. Suppose you want to give some characters a code length of 3 (and you haven't given any character a code yet). In that case you've got 8 (2 to the power 3) possibilities. Seven codes can be immediately allocated to characters. The last one depends on how many more characters you have left to code. If you only have one left to code, the last code will do, but if you have to code more than one, you must use the last code to indicate that an extension follows. The extension has the same structure. This has to be continued until all different characters in the text are given a code.

For example you've the characters '`a`', '`e`', '`i`', '`o`', '`u`' and '`y`'. You choose to give the first three characters a code with length 2 and the rest an equal length. So '`a`' becomes '00', '`e`' becomes '01' and '`i`' becomes '10'. The other characters have to be an extension of '11'. Then '`o`' becomes '1100', '`u`' becomes '1101' and '`y`' becomes '1110'. The code '1111' is free now. If 7 characters instead of 6 characters should be given a code this last code would be sufficient. If more than 7 characters should be given a code, the last code would be extended and so on.

You only have to code the characters which occur in the text. You don't have to give the code table itself.

You must write a program that reduces a given text by recoding each character and give the minimum total filelength in bits.

Input

The first line of the input file contains the number of problems. A problem starts with the number of lines n to follow, followed by n lines. A line consists of characters and the characters '`A`'..'`Z`', '`a`'..'`z`', `space`, '`.`', '`,`' , '`-`' and '`$`' are allowed. The character '`$`' will always be at the end of a line and cannot occur anywhere else. This character has to coded instead of the real end of line mark.

Output

For each problem in the input your program should output the minimal number of bits to code the given text.

Sample Input

```
2
3
Hello Contestant,$
Please write a program which gives$
the text Hello world.$
1
To be or not to be, that is the question.$
```

Sample Output

```
335
167
```