It's Easy

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S, you can do the following operation **at most** once:

Choose a substring from the string S and reverse it. For example, if S = "bbabcbb" and you choose to reverse the substring [3,5], then the string S will be "bbcbabb".

The target is to maximize the number of occurrences of the string "ab" as a subsequence in the string S.

A substring of a string is a contiguous subsequence of that string. So, string "bcb" is a substring of string "aabcbaa", but string "bca" is not.

A subsequence of the string t is such a sequence that can be derived from the string t after removing some (possibly zero) number of letters without changing the order of the remaining letters. For example, the string "baacbc" contains the subsequences "abc", but doesn't contain a subsequence of "bca".

Input

The first line of the input contains an integer t ($1 \le t \le 10$) represents the number of test cases.

The first line of each test case contains an integer n represents the size of the string ($1 \le n \le 200$).

The second line of each test case contains the string S consisting of lowercase Latin letters.

Output

For each test case print one integer, the maximum number of subsequences of "ab" in the input string after performing at most one operation.

Example

standard input	standard output
3	4
5	6
aacbb	23
7	
acbabbe	
16	
abcbccabdaebbabb	

Note

In the **first** test case:

The string "aacbb" has 4 subsequences of "ab": (1,4), (1,5), (2,4), and (2,5), so it's not optimal to reverse any substring.

In the **second** test case:

The string "acbabbe" has 5 subsequences of "ab", but if we reverse the substring [3,4] the string will become "acabbbe" which has 6 subsequences of "ab".