Your librarian friend is planning to digitize the catalog of the local library and needs your help. He gives a list of strings to be shown in the online catalog and asks you to sort them in a dictionary-like order. You decide to help him by writing a program that uses the Radix Sort algorithm to sort the given strings.

Problem

In this exam, you are asked to sort the given string array *arr* with Radix Sort in ascending or descending order depending on the boolean variable *ascending* and return the number of iterations done in the loops of the Counting Sort algorithm (you need to use Counting Sort as a subroutine in the Radix Sort). *size* is the number of elements in the string array.

```
int radix_string_sort(std::string *arr, int size, bool ascending);
```

Constraints and Hints:

- Array elements will be strings each of which can contain only the characters as uppercase English letters (i.e. from 'A' to 'Z').
- It will be easier to follow the iteration count if you implement your solution by modifying the pseudocodes given in your book.
- Different from the Radix Sort algorithm in your book, it is not guaranteed that the strings in the array will always have the same length. (*Hint:* You can use an extra bucket during Counting Sort to handle strings with different lengths.)
- Different than the algorithm for Counting Sort in your book, initialize the count array as *int* C = new int[k]* and use the fourth loop for copying the array back. That means, you shouldn't count iterations during initialization, but you should count iterations during copying array back. Otherwise, the return value of the function (as the number of iterations) will not be evaluated as correct.
- You should count loop iterations in four different loops.
- *Ascending order means dictionary order*. For example, when ascending=true, OGUZ must reside in the result array before OGUZHAN. If ascending=false, it is the other way around.