RADIX SORT OF STRINGS

@ HTTP://WWW.MARTINBROADHURST.COM/RADIX-SORT-OF-STRINGS-IN-C.HTML

You can use radix sort to sort any kind of data that can be seen as a sequence of symbols. This includes strings, where the symbols are the individual letters, and numbers, where the symbols are the digits. Radix sort works by having a bucket for each value that a symbol can have, and putting data items into buckets according to the value of each symbol in the item in turn, starting with the rightmost. At the end of the sort, the items will be in **order of length**, and then in **lexicographic order** within each length class. This is called **shortlex order**.

An example will make this clearer. Supposing we have the set of words in the string "**The quick brown fox jumps over the lazy dog**". Once sorted by radix sort they will be in the following order:

```
The
dog
fox
the
lazy
over
brown
jumps
quick
```

Notice that it is by length first, and then lexicographic.

The algorithm requires a bucket (Linked List) for each value that a character will have, so we have 256 buckets in order to accommodate all of ASCII characters.

We proceed by taking each character in the strings in turn, starting with the rightmost, so the first iteration of the algorithm will process the last letters of each word, which are **[e, g, x, e, y, r, n, s, k]**. We put the words into the buckets that correspond to these letters, so the buckets look like this:

```
[101] The -> the
[103] dog
[107] quick
[110] brown
[114] over
[115] jumps
[120] fox
[121] lazy
```

The bucket numbers are the ASCII codes of the characters. I have omitted empty buckets. Notice that "The" and "the" go in the same bucket because they both end in "e". The buckets are implemented as linked lists so they can hold multiple entries.

Now, we collect the words together in bucket order, so we have a single list like: this:

```
The -> the -> dog -> quick -> brown -> over -> jumps -> fox -> lazy
```

We now consider the second letters of each word from the right, which are **[h, h, o, c, w, e, p, o, z]**, and repeat the process of putting the words in buckets:

```
          [99] quick
          [101] over
          [104] The -> the
          [111] dog -> fox
          [112] jumps
          [119] brown
          [122] lazy
```

Collecting the words together, we get:

```
quick -> over -> The -> the -> dog -> fox -> jumps -> brown -> lazy
```

The third letters from the right are **[i, v, T, t, d, f, m, o, a]**, so the buckets are:

```
          [84] The
          [97] lazy
          [100] dog
          [102] fox
          [105] quick
          [109] jumps
          [111] brown
          [116] the
          [118] over
```

The collected words are now:

```
The -> lazy -> dog -> fox -> quick -> jumps -> brown -> the -> over
```

"The", "dog", "fox", and "the" only have three letters, so they have no fourth letter from the right. That's fine, because we consider this letter to be 0, and we have a 0 bucket, so in the fourth iteration those four words end up in the 0 bucket, while the remaining words are bucketed according to their fourth letter from the right:

```
          [0] The -> dog -> fox -> the
          [108] lazy
          [111] over
          [114] brown
          [117] quick -> jumps
```

The collected list is now:

```
The -> dog -> fox -> the -> lazy -> over -> brown -> quick -> jumps
```

Now we can see that we are making progress. The four three-letter words “The”, “dog”, “fox”, and “the” are now at the head of the list, and what’s more, they are in lexicographic order.

On the fifth iteration, “lazy” and “over” end up in the 0 bucket, and once again, they are in lexicographic order, and because the items were added to the buckets in order of the result of the previous iteration, they go after the four three letter words.

```
          [0] The -> dog -> fox -> the -> lazy -> over
          [98] brown
          [106] jumps
          [113] quick
```

Because the longest word is 5 letters, we have finished after 5 iterations. There is no need to do a sixth iteration to put the two five-letter words “jumps” and “quick” into the 0 bucket, because they are already sorted by the fifth letter, so we just collect the items from the buckets in the normal way and we are done:
Here is the collected list:

```
The -> dog -> fox -> the -> lazy -> over -> brown -> jumps -> quick
```

So you can see that the number of iterations of the algorithm is the number of characters in the longest word.

Good Luck!