Code Jam 2009 - Round 1B

Analysis: The Next Number

Let **x** be our input. We want to find the next number **y**. We denote L(*s*) to be the length of *s*, i.e., the number of digits in *s*.

Case 1. If all the digits in *x* are non-increasing, for example *x* = 776432100, then *x* is already the biggest one among the numbers in the list with L(*x*) digits. The next number, *y*, must have one more digit, that is, one more 0. Actually *y* must be the smallest one with L(*x*)+1 digits. To get *y*, we put the smallest non-zero digit in front, and all the other digits are put in non-decreasing order.

Case 2. Otherwise, *x* can be written as the concatenation *x* = *ab*, where *b* is the longest non-increasing suffix of *x*, so *d*, the last digit of *a* is smaller than the first digit of *b*. Let *d'* be the smallest among all the digits in *b* that are bigger than *d*.
Because *b* is non-increasing, *x* is the biggest number among those who has L(x) digits and starts with prefix *a*. The first L(*a*) digits of *y* must be bigger than *a*. The smallest we can do is to replace *d* with *d'*, and then for the rest digits, we arrange them in non-decreasing order.
Let us do another example. *x* = 134266530. Then *a* = 1342, *b* = 66530, *d* = 2, and *d'* = 3. The next number is *y* = 134302566.

In fact, we can unify the two cases above. Since the number of 0's is not restricted, we can just imagine there is one more 0 in the beginning of *x*, thus Case 1 is reduced to Case 2.

The above described is actually exactly the procedure to get the next permutation of a finite sequence in certain languages. Below is a solution that is essentially one line in C++. From the author:

```
deque<char> f;
...
f.push_front('0');
next_permutation(f.begin(), f.end());
if (f.front() == '0') f.pop_front();
```