XAR lab recently develops a new computer for data compression — "XAR08". Each time, XAR08 will get an integer sequence from input, and output it after compression.

XAR08 is composed of several 8-bit binary storage units. Each storage unit can store an 8-bit unsigned integer and support 4 directives. A program in XAR08 is a directive sequence composed of these 4 directives as follows:

`X` $n$ the integer in each storage unit XOR $n$, $0 \leq n < 256$,

Equivalence: $V = V\hat{}n$

`A` $n$ add n to each storage unit and mod 256, $0 \leq n < 256$,

Equivalence: $V = (V + n)\%256$

`R` $n$ rotate each storage unit n-bit binary left, $0 \leq n < 8$,

Equivalence: $V = (((V >> (8 - n))|(V << n))\&0\text{xFF})$

`E` $n$ the program ends, $0 \leq n < 256$, ignore the value of $n$. Every program should end with this directive.

Each time, XAR08 gets an integer sequence with the length of $N$ from input. These $N$ integers will be stored in the first $N$ storage units in order (The number of storage units is enough). After compression, the value in these $N$ storage units will be sent to output in the same order.

XAR08's data compression operation is based on a transformation $f$: Transform the input sequence (all elements are different) $D = (d_0, d_1, \ldots d_{n-1})$ to the sequence $(0, 1, \ldots, n-1)$, i.e. $f(d_i) = i(0 \leq i < n)$.

Your task is, for each input sequence, write an XAR08 program composed of the above four directives to implement the transformation $f$. XAR08 is still in research stage, so it can only execute a program with no more than $40,000$ directives.

## Input

Input contains several cases. The first line in each case contains an integer $n$ ($n \leq 128$), which is the length of sequence $D$, followed by a line of $n$ different integers, $d_0, d_1, \ldots, d_{n-1}$, $0 \leq d_i < 128$.

The last case is followed by a line containing a zero.

## Output

For each case, the first line outputs '`Case ?:`'. If exists a XAR08 program composed of no more than $40,000$ directives, output the program from the second line. Otherwise output '`Impossible!`' (quotes for clarity) in the second line.

Don't print any extra spaces or blank lines.

## Sample Input

```
1
123
3
2 1 0
0
```

## Sample Output

```
Case 1:
X 123
E 0
Case 2:
X 3
A 255
E 0
```