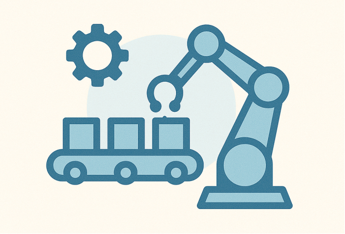
I. Production Order Reconstruction

in Smart Manufacturing Line

Constraint: Time Limit: 1 seconds, Memory: 64MB



Background

The robot manufacturing company *FuturaMech* operates an automated assembly line consisting of **n stages (**1≤n≤100000**)** numbered from 1 to n.

Each stage has a **priority level** (determined by its index: a stage with a smaller index has lower priority) and different **completion times**.

Due to a system failure, the file recording the completion order of stages has been lost. However, the control department has retained one important piece of information:

For each stage i (1 ≤ i ≤ n), there are exactly a_i stages with **lower priority** (index j < i) that were **completed earlier** than stage i ($-10^9 \leq a_i \leq 10^9$). Data is considered valid if and only if: $0 \leq a_i \leq i-1$ for all $i = 1,2,\dots,n$

Requirements

Write a program to **reconstruct the completion order** of the stages based on the given information.

Note: If the data is invalid (violates logical constraints), the program must detect and report an error.

Input/Output Format

Input	Output
▪ Line 1: Integer **n** - number of stages ▪ Line 2: **n** integers $a_1, a_2, \dots, a_n$	▪ **If data is valid:** Print **n** integers $p_1, p_2, \dots, p_n$ which is a permutation of 1, 2, ..., n, where: p_i is the stage completed at **position i** (in chronological order) ▪ **If data is invalid:** Print **-1**

Examples

Testcase 01

Input	Output
5 0 1 1 0 0	5 4 1 3 2

Output order q = [5, 4, 1, 3, 2].
Positions: pos(5)=1, pos(4)=2, pos(1)=3, pos(3)=4, pos(2)=5.

For each stage i, a_i counts how many **lower-priority** stages j<i finish **earlier** (i.e., pos(j) < pos(i)):

- i=1: lower-priority set is empty ⇒ a_1=0.
- i=2: lower-priority {1}; pos(1)=3 < pos(2)=5 ⇒ count 1 ⇒ a_2=1.
- i=3: lower-priority {1,2}; only 1 is earlier (3 < 4) ⇒ count 1 ⇒ a_3=1.
- i=4: lower-priority {1,2,3}; all appear after pos(4)=2 ⇒ count 0 ⇒ a_4=0.
- i=5: lower-priority {1,2,3,4}; all appear after pos(5)=1 ⇒ count 0 ⇒ a_5=0.

All counts match a = [0, 1, 1, 0, 0], so the output is valid.

Testcase 02

Input	Output
3 0 2 1	-1

Stage 2 has only one lower-priority stage ({1}), so the maximum possible value for a_2 is 1 (when stage 1 finishes before stage 2). Since a_2 = 2 > 1, this violates the bound $a_i \leq$ i−1, hence no valid completion order exists.