Találd ki a permutációt! (findperm-pp)

✍ Ez egy interaktív feladat, amiben a programodnak az értékelő rendszerrel kell kommunikálnia. Felváltva írj ki üzeneteket a standard kimenetre és olvasd be a következő bemenetet a standard bemenetről!

Adott az $1, 2, \ldots, N$ számoknak egy ismeretlen, $P = [P_1, P_2, \ldots, P_N]$ permutációja [1].

1. ábra. A permutáció a falba bújt el előled.

Kérdéseket tehetsz fel (i, j) alakban, ahol $1 \leq i, j \leq N$. Az i és j számoknak nem kell különbözőnek lenniük. Az értékelő program a kérdésedre a $(P_i$ `AND` $P_j)$ [2] érték **legmagasabb helyi értékű bit** indexével válaszol.

A legmagasabb helyi értékű bit indexén az x kettes számrendszerbeli alakjának legmagasabb hatványkitevőjét értjük. Például, $5 = 101_{(2)}$ esetén ez az érték 2, hiszen az 5 kettes számrendszerbeli alakjában a $4 = 2^2$ a legnagyobb tag. Ha P_i `AND` P_j értéke 0, az értékelő program `-1`-gyel válaszol.

Feladatod a legfeljebb 1000 számból álló P permutációt kitalálni maximum 30 000 kérdés használatával.

☞ Az értékelő rendszerből letölthető csatolmányok közt találhatsz `findperm.*` nevű fájlokat, melyek a bemeneti adatok beolvasását valósítják meg az egyes programnyelveken. A megoldásodat ezekből a hiányos minta implementációkból kiindulva is elkészítheted.

Bemenet

A bemenet első sora az N számot tartalmazza, a permutáció hosszát. Miután ezt beolvastad, a kérdéseket a standard kimenetre az alábbi formátumban írhatod ki:

```
? i j
```

ahol i és j a (i, j) kérdést írják le.

[1] Az $1, 2, \ldots, N$ számok permutációja egy olyan N hosszúságú sorozat, melyben az $1, 2, \ldots, N$ értékek mindegyike pontosan egyszer szerepel.

[2] Az x és y számokból a bitenkénti ÉS (`AND`) művelet egy olyan számot képez, amiben a két szám közös bitjei szerepelnek. Például, ha $x = 11 = 1011_{(2)}$ és $y = 13 = 1101_{(2)}$, akkor x `AND` $y = 1001_{(2)}$.

Amennyiben i és j érvényes indexek ($1 \leq i, j \leq N$) és a feltett kérdések száma a most feltett kérdéssel nem haladja meg a 30 000-et, az értékelő program a standard bemenetre kiírja a választ. Egyéb esetben a válasz `-100`, és az értékelő program `Helytelen válasz` eredményt ad.

Ha kitaláltad a permutációt, írd ki az alábbi formátumban:

`!` P_1 P_2 $\ldots$ P_N

A megoldás kiírása nem számít kérdésfeltevésnek.

Miután kiírtad a helyesnek gondolt permutációt, vagy `-100`-at kaptál válaszul egy kérdésre, a programod azonnal álljon le; ellenkező esetben `Időtúllépés` értékelést kaphatsz.

Korlátok

- $1 \leq N \leq 1000$.

Pontozás

A megoldásodat sok különböző tesztesetre lefuttatjuk. A teszteseteket **különállóan** pontozzuk, a végső pontszámot a különböző tesztekre elért pontszámok összege adja. A kitalálandó permutáció minden tesztesetben **előre rögzített**, a programod válaszainak hatására az értékelő program ezt nem változtatja meg.

Jelölje Q a programod által az adott tesztesetre feltett kérdések számát. A pontszámod az alábbiak szerint kerül kiszámításra:

- Ha $Q > 30\,000$, a tesztre 0 pontot kapsz;
- Ha $25\,000 < Q \leq 30\,000$, az adott tesztre kapható maximális pontszám 20%-át kapod meg;
- Ha $20\,000 < Q \leq 25\,000$, az adott tesztre kapható maximális pontszám 40%-át kapod meg;
- Ha $15\,000 < Q \leq 20\,000$, az adott tesztre kapható maximális pontszám 60%-át kapod meg;
- Ha $Q \leq 15\,000$, a tesztre kapható pontszám egészét megkapod.

Példák

input	output
5 2 1 -1 0	 ? 1 5 ? 2 3 ? 4 3 ? 4 4 ! 4 3 2 1 5

Magyarázat

Tegyük fel, hogy a $P = [4, 3, 2, 1, 5]$ permutációt kell kitalálnod.

Az első kérdés a $P_1 = 4$ és $P_5 = 5$ legnagyobb indexű helyi értékére kérdez rá, ami 2.

A második kérdés a 3 és 2 értékekre kérdez rá, amire a válasz 1.

2 `AND` $1 = 0$, ezért a harmadik kérdésre a válasz `-1`.

Végül a negyedik kérdésre a válasz 0, hiszen 1 `AND` $1 = 1$.

Ez az interakció példaként szolgál, hogy bemutassa a kérdések feltevésének, az értékelő válaszának és a megoldás kiírásának működését. **Nem feltétlenül** jelent egy helyes módszert a permutáció kitalálására.