Tree Search

У вас есть корневое бинарное дерево, состоящее из N вершин. Вершины пронумерованы от 1 до N, при этом корень имеет номер 1. У каждой другой вершины есть единственный родитель в дереве. Дерево является бинарным, то есть каждая вершина может быть родителем не более двух других вершин.

Одна из вершин является особенной. Вы пытаетесь её угадать. Вы можете задавать вопросы следующего вида: "Находится ли особенная вершина в поддереве вершины x"?

Вершина y находится в поддереве вершины x тогда и только тогда, когда кратчайший путь между y и 1 проходит через вершину x. Обратите внимание, что вершина x также является частью своего собственного поддерева.

Вам разрешено задать этот вопрос не более 35 раз. Затем вы должны предоставить свою догадку.

Implementation Details

Вам следует реализовать следующую функцию:

```
int solve(int N, std::vector < int > p)
```

- N: количество вершин
- p содержит ровно $N-1$ элементов, описывающих дерево: вершина $p[i]$ (где $1 \leq p[i] \leq i+1$) является родителем вершины $i+2$ для каждого $0 \leq i \leq N-2$
- Ни один элемент в p не встречается более двух раз
- *Эта функция должна возвращать номер особой вершины*
- *Эта функция вызывается ровно один раз*

Вышеуказанная функция может делать вызовы следующей функции:

```
int ask(int x)
```

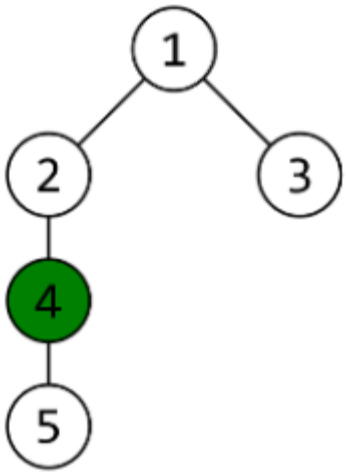
- x: номер вершины
- $1 \leq x \leq N$
- *возвращает* 1*, если особая вершина находится в поддереве вершины* x *и* 0 *в противном случае*

Example

Рассмотрим следующий вызов:

```
solve(5, [1, 1, 2, 4])
```

Дерево состоит из рёбер (1,2), (1,3), (2,4) и (4,5).



Ваша программа сделала вызов

```
ask(4)
```

который вернул 1.

После этого ваша программа сделала вызов

```
ask(5)
```

который вернул 0.

Ваша программа пришла к выводу, что вершина 4 является особенной и вернула 4.

Constraints

- $2 \leq N \leq 100\ 000$

Subtasks

1. (20 баллов) $N \leq 35$
2. (30 баллов) $p[i] = i + 1$ для каждого $0 \leq i \leq N - 2$
3. (15 баллов) $p[i] = \lfloor i/2 \rfloor + 1$ для каждого $0 \leq i \leq N - 2$

4. (35 баллов) Дополнительных ограничений нет.

Sample Grader

Грейдер читает входные данные в следующем формате:

- строка 1: N
- строка 2: $p[0], p[1], ..., p[N-2]$

Грейдер выводит каждый вопрос в следующем формате:

- строка 1: $?\ x$

Грейдер читает каждый ответ в следующем формате:

- строка 1: y

Грейдер выводит предположение в следующем формате:

- строка 1: $!\ x$