Evolution

Damyoo is studying the evolutionary history of $N = 256$ species, numbered from 0 to $N - 1$. Species 0 is the root of the ancestry tree. For every species i other than species 0, its direct ancestor is species $P[i]$.

A species is considered an ancestor of itself. The **most recent common ancestor** of two species u and v is their common ancestor that is farthest from species 0. Equivalently, it is their lowest common ancestor in the rooted tree.

Damyoo divides the analysis between two isolated research stations, operated by Alice and Bob. Before the experiment begins, Alice and Bob may agree on any deterministic protocol. During the experiment, they cannot communicate except through binary labels produced by Alice.

Alice first receives the complete ancestry tree and assigns one nonempty binary label to every species. Later, Bob receives only the labels of two species. He is not given the ancestry tree or the numbers of the two species, and must determine the number of their most recent common ancestor.

Your solution must always answer correctly while minimizing the maximum length of a label produced by Alice.

Implementation Details

You should implement the following procedures:

```
vector<vector<bool>> Alice(int N, vector<int> P)
```

- N: the number of species.
- P: an array of length N. $P[0] = -1$, and for every $1 \leq i < N$, $P[i]$ is the direct ancestor of species i.
- This procedure is called exactly once during the first run of the grader.
- It should return an array L of length N. For every $0 \leq i < N$, $L[i]$ is the label assigned to species i.
- Every element of L must be nonempty and must have length at most $2\,048$.

```
int Bob(int N, vector<bool> U, vector<bool> V)
```

- N: the number of species.
- U, V: labels assigned by the same call to `Alice` to two species in the same ancestry tree.
- This procedure is called between 1 and 100 times during the second run of the grader.
- It should return the number of the most recent common ancestor of the two species whose labels are U and V.

For each test case, the grader runs your program exactly twice:

- During the first run, `Alice` is called exactly once and `Bob` is not called. The returned labels are stored by the grading system.
- During the second run, `Bob` is called at most 100 times and `Alice` is not called. Your program can store and retain information across successive calls to `Bob`.
- The only information available during the second run from the first run consists of the labels passed as arguments to `Bob`. In particular, information stored in global or static variables during the first run is not available during the second run.

Your code must include the header file `evolution.h`. You should not implement a `main` function, and your code must not read from the standard input or write to the standard output.

Constraints

- $N = 256$.
- $P[0] = -1$.
- $0 \le P[i] < N$ and $P[i] \neq i$ for every $1 \le i < N$.
- The parent relations form a rooted tree whose root is species 0.
- `Bob` is called between 1 and 100 times for each test case.
- Every label passed to `Bob` was returned by the corresponding call to `Alice`.

Subtasks and Scoring

Subtask	Score	Additional Constraints
1	5	$P[i] = i - 1$ for every $1 \le i < N$.
2	95	No additional constraints.

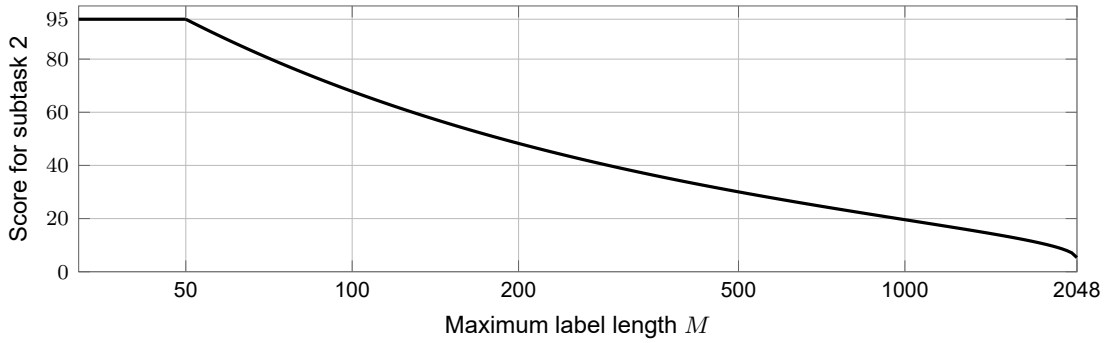
For subtask 2, let M be the maximum length of any label returned by `Alice` over all its test cases.

A subtask receives 0 points if any returned array or label is invalid, or if any answer returned by `Bob` is incorrect.

If subtask 2 is solved correctly, its score is determined only by M. If $M > 2\,048$, the score is 0. Otherwise, the score for subtask 2 is

$$5 + 90 \min\left(1, \sqrt{\frac{2\,048/M - 1}{2\,048/50 - 1}}\right).$$

The score is rounded to two decimal places.



Example

Consider an ancestry tree satisfying $P[1] = P[2] = 0$, $P[3] = P[4] = 1$, and $P[i] = 0$ for every $5 \leq i < N$.

Suppose that a valid call to `Alice` assigns the following two labels to species 3 and 4, respectively. Each brace-enclosed list denotes a `vector<bool>`.

```
{0, 0, 0, 1, 1}
{0, 0, 0, 0, 0, 1, 0, 0}
```

The grader may then make the following call:

```
Bob(
    N,
    {0, 0, 0, 1, 1},
    {0, 0, 0, 0, 0, 1, 0, 0}
)
```

This call must return 1, because the most recent common ancestor of species 3 and 4 is species 1.

Sample Grader

The sample grader calls `Alice` and `Bob` in the same run, which is different from the actual grader.

Input format:

```
N
P[1] P[2] ... P[N-1]
Q
A[0] B[0]
A[1] B[1]
...
A[Q-1] B[Q-1]
```

For every $0 \leq j < Q$, $A[j]$ and $B[j]$ are the numbers of the two species in the j-th query.

Output format:

```
R[0]
R[1]
...
R[Q-1]
```

Let L be the array returned by `Alice(N,P)`. For every $0 \leq j < Q$, the sample grader sets $R[j]$ to the value returned by `Bob(N,L[A[j]],L[B[j]])`. Only the values $R[0], R[1], \ldots, R[Q-1]$ are printed.

Note that you can use the sample grader with any value of N.