Problem Speedrun

C++ header `speedrun.h`

Marcel started doing speedruns, in hopes to get a new WR (World Record). Sadly, he cannot master the newly discovered strats, so he has to find a way to gain an unfair advantage.

The game he's trying to get good at is called *Tree Souls III*, and he's trying to get a WR in the *full-tree%* category.

The game takes place, as the name implies, on a tree (i.e. a graph with N vertices, numbered from 1 to N, and $N-1$ edges, such that there are no cycles).

The game works as follows: the character is placed in an arbitrary node inside the tree. He can choose an integer x, and try to walk from the node he's in to the node x. If there is an edge between x and the node he's standing in, he will move to x, otherwise nothing happens. His speedrun is valid if he visits each node at least once.

Now, here comes the cheating part: since he doesn't know the edge-teleportation-glitch, he is going to set the seed for his world, therefore, he will know how the tree looks like before starting the run. If he just memorizes the tree, it will be too obvious that he's cheating, so he will be instantly banned from the speedrunning community, therefore he will just put some hints in each node (by tampering with the code). A hint is a binary string of size l (l is the same for every hint in each node). When he sits on a node, he can read the hint assigned to the node he's currently in.

Interaction Protocol

This is a two-interaction problem!

Your code will be run twice, one run for the first interaction (the *hint-setting phase*), and one run for the second interaction (the *speedrunning phase*).

You must implement the following functions (and not `main`).

```
void assignHints(int subtask, int N, int A[], int B[]);
void speedrun(int subtask, int N, int start);
```

Using the following callable functions:

```
void setHintLen(int l);
void setHint(int i, int j, bool b);
int getLength();
bool getHint(int j);
bool goTo(int x);
```

Hint-setting phase. This is the first run of your code. In this phase, the committee's code will call the `assignHints` function exactly once. It will give it *subtask*, which is the index of the given subtask, and N as parameters, and it will pass the edges of the tree in A and B as follows: for each $i = 1, \ldots N-1$, there will be an edge between $A[i]$ and $B[i]$. You must then call the `setHintLen` function exactly once, giving it the length of hints you have decided to use as a parameter. After you have called `setHintLen`, you may then call the `setHint(i, j, b)` function multiple times. This sets bit `j` of the hint for node `i` to `b`. The bits are indexed from 1 to l. The hints are initially filled with zeroes. In this phase you must not call the `getLength`, `getHint` or `goTo` functions. Your code should exit from `assignHints` when you are done assigning hints.

Speedrunning phase. This is the second run of your code. In this phase, the committee's code will call the `speedrun(subtask, N, start)` function exactly once, telling it the node `start` in which the speedrunner begins, as well as the value of N. Additionaly, you also receive the index of the subtask

through the parameter *subtask*. You can then call the `getLength()` function, which returns l, the length of the hints set by your program in the first phase, the `getHint(j)` function which tells you bit j of the hint for the node you are currently at, and the `goTo` function. If the parameter given to `goTo` is the index of a node with an edge to the current node, then the function returns `true` and you move to that node. Otherwise, it returns `false` and you do not move from the current node. In this phase you must not call the `setHint` or `setHintLen` functions. Your code should eventually exit from the `speedrun` function after you have visited all the nodes in the tree.

Restrictions

- $1 \leq N \leq 1\,000$
- Let Q be the number of calls to $goTo$ which returned $false$. In what follows we outline constraints for l and Q that must be respected in order to earn score for each subtask.

Subtask 1 (21 points)

- $l \leq N$
- $Q \leq 2000$
- The score for the subtask will be 21 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 2 (8 points)

- $l \leq 20$
- $Q \leq 2000$
- The tree is a star; i.e. there is a node x ($1 \leq x \leq N$) which is connected to every other node.
- The score for the subtask will be 8 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 3 (19 points)

- $l \leq 20$
- $Q \leq 2000$
- The degree of each node is at most 2.
- The score for the subtask will be 19 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 4 (12 points)

- $l \leq 316$
- $Q \leq 32000$
- The score for the subtask will be 12 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 5 (40 points)

- $Q \leq 2000$
- The score s of each test is computed as follows. If $l > 40$, then $s = 0$. If $l \leq 20$, then $s = 40$. Otherwise, $s = 60 - l$. That is, if $l = 40$, you will receive half the points for the test, and if $l \leq 20$, you will receive full score for the test. The score for the subtask will be the minimum of the values s over all tests in the subtask.

Interaction example

In the first interaction (hint-setting phase), the contestant's function will be called as follows:

```
assignHints(
  /* subtask = */ 1,
  /* N       = */ 5,
  /* A       = */ {-, 1, 2, 3, 3},
  /* B       = */ {-, 2, 3, 4, 5});
```

This function, in turn, might choose to set $l = 2$:

```
setHintLen(/* l = */ 2);
```

And then to leave all bits of the hints set to 0, except for bit 1 of node 2, which it sets to 1:

```
setHint(
  /* i = */ 2,
  /* j = */ 1,
  /* b = */ 1);
```

Then, it exits. The first instance of the contestant's program will now be terminated, hence any data being stored in memory by this program will be lost. By now, the hints are `"00"` for every node, except node 2, which has the hint `"10"`.

In the second interaction (speedrunning phase), the contestant's function will be called:

```
speedrun(
  /* subtask = */ 1,
  /* N       = */ 5,
  /* start   = */ 1);
```

In reply, the function begins speedrunning the tree:

```
getLength(); // = 2
getHint(1);  // = 0
getHint(2);  // = 0
goTo(2);     // = true
getHint(1);  // = 1
getHint(2);  // = 0
```

At this point, you were initially in node 1, found out that $l = 2$ and that the hint of node 1 is `"00"`, then successfully moved to node 2 and found out that the hint of node 2 is `"10"`. Execution can continue as follows:

```
goTo(2);     // = false (you can not go from node 2 back to itself)
goTo(5);     // = false (there is no edge from node 2 to node 5)
goTo(3);     // = true
goTo(4);     // = true
goTo(3);     // = true
goTo(5);     // = true
```

At this point all nodes have been reached, and so the `speedrun` function may exit. The value of $Q = 2$, because 2 calls to `goTo` have returned `false`.