Lollipops, Sweets and Chocolates: GPS

Type	Time limit	Memory limit
Interactive	2 seconds	256 MB

Statement

It's holiday season at the Australian Ice-Cream and Oreo Crescent (AIOC), and as a new marketing strategy, its popular candy store franchise Lollipops, Sweets and Chocolates (LSC) have come up with a new device, the Gumball Positioning System (GPS).

Your town can be modelled as a network of N intersections (numbered from 1 to N) and M two-way roads of equal length. There will never be two roads between the same pair of intersections, and no road will connect an intersection to itself.

The GPS is a revolutionary device that allows you to input the intersection that you are at, and a list of your favourite LSC stores. The GPS will then return the closest such store to your current location (by road), and if multiple exist, the **first** such store in your list is returned. There is a store at each intersection.

Having found yourself hopelessly lost in town, you decide to reconstruct a map of the town's M roads and N intersections. However, your trusty GPS is low on charge, so you cannot use it more than Q times.

Subtasks and Constraints

For all subtasks:

- $2 \leq N \leq 400$
- $N - 1 \leq M \leq 600$
- The graph will be connected.
- Each road will connect two different intersections.
- No two roads connect the same pair of intersections.
- The grader is non-adaptive. In other words, the roads have already been decided before any queries are made to the grader.

Subtask	Points	Maximum N	Q	Additional constraints
1	15	10	2000	None
2	37	400	2000	None
3	25	400	797	$M = N - 1$. That is, the graph is a tree.
4	23	400	1000	None

Input/Output

This task has no input or output files. Instead, your solution must interact with the functions in the header file "gps.h". The provided functions are described in detail in the next section. **Do not output anything to stdout, or you will receive 0% of the points for the test case.**

Implementation

You **must not** implement a main function. Instead, you should `#include "gps.h"` and implement the function:

```
void reconstruct(int s, int n, int m, int q);
```

where:

- `s` is the subtask number
- `n` is the number of intersections
- `m` is the number of roads
- `q` is the maximum number of queries

In your implementation, you may call the following functions:

```
int findBest(int location, std::vector<int> intersections);
```

where:

- `location` is the number of the intersection representing the location you want to give the GPS
- `intersections` is the list of the numbers of the favourite intersections you want to give the GPS. This list must be non-empty and contain **distinct elements**.
- `findBest` returns the first intersection in your list that has minimal distance to your intersection.

You may assume that this function takes $O(N)$ time.

To report roads to the grader, you may call the following function:

```
void reportRoad(int u, int v);
```

To report that there is a road between intersection u and intersection v. Obviously, $1 \leq u \leq N$ and $1 \leq v \leq N$

Please note that if your input to `findBest` or `reportRoad` does not conform to the above requirements, or if you report a road that you have already reported, or if you report a road that does not exist, then you will receive 0% of the points for the test case.

Experimentation

The source files `gps.h` and `grader.cpp` are provided for you to experiment on your machine. Please note that the grader used may have different behaviour to the provided grader.

Compile your solution with:

```
g++ -std=c++11 -O2 -Wall -static gps.cpp grader.cpp -o gps
```

Sample Grader

The sample grader reads in four integers: `S N M Q`: the subtask number, number of intersections, number of roads and max number of queries respectively.

It will then read M more lines. From the i-th of these lines, the sample grader will read two integers `u v`, representing that there is a road from intersection u to intersection v.

The sample grader will then call the function `reconstruct` and output `Correct` if your program reported all the roads or the appropriate error message if your program terminated before it output all the roads, used more than Q queries, or reported an incorrect road. If correct, the sample grader will also report the number of queries used.

Sample Grader Input and Sample Session

```
1 5 5 2000
5 1
3 1
1 2
2 3
2 4
```

Grader	Program	Description
reconstruct(1, 5, 5, 2000)		The grader initialises your program.
	findBest(5, [3, 2, 4])	The program calls findBest from intersection 5, and stores 3, 2 and 4.
Grader returns 3		2 and 3 are both of distance 2 from 5, but 3 occurs earlier in the list, so 3 is returned.
	findBest(2, [4, 2])	The program calls findBest.
Grader returns 2		2 is closer to 2 than 4 is to 2, so 2 is returned.
	reportRoad(1, 5)	The program reports that there is a road from intersection 1 to intersection 5. This is correct.
	reportRoad(1, 4)	The program reports that there is a road from intersection 1 to intersection 4. This is not correct. The grader terminates and gives a score of zero. What. b. Shame!