Chain of Atoms

PROBLEM

Scientists are investigating a special kind of molecule. It is known that the molecule consists of *N* different atoms, labeled from 1 to *N*, and the structure of the molecule is a linear chain. The scientists are equipped with a distance meter. At one time, this instrument can measure the distance between any two atoms in the molecule. The distance, reported by the instrument, is the absolute value of the difference of the two atom positions in the chain. If you access no atom for more than three times no damage is done to the molecule. If you access one or more atoms four times the molecule gets damaged. If you access one or more atoms five times the molecule gets destroyed.

You are to write a program that discovers the complete structure of the molecule by determining the sequence of the atoms in the molecule.

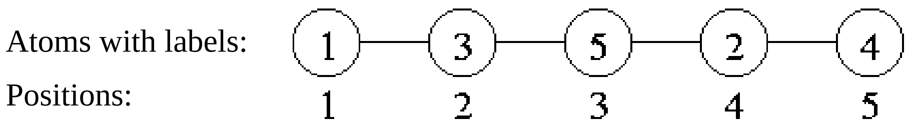

Figure 1

LIBRARY

To operate the distance meter, you are given a library `meter` with three operations:

- `Size`, to be called once at the beginning, without arguments; it returns the value of *N*. `Size` must be called before any call to `Span`.
- `Span`, to be called with two atom labels as arguments; it returns the distance between the atoms.
- `Answer`, to be called at the end, must be used to submit your solution. `Answer` has two integer arguments, *i* and *x*, and it tells that the label of the atom at position *i* in the molecule is *x*. For each *i* ($1 \le i \le N$), `Answer` must be called exactly once in *ascending* order of *i*. There are always two symmetric solutions, and you can submit any one of them.

The dialogue between your program and the library procedures is logged in the text file `chain.out`. This log file also contains your answer and the error message in case of an error.

Instruction for Pascal programmers: include the import statement

```
uses meter;
```

in the source code.

Instructions for C/C++ programmers: use the instruction

```
#include ”meter.h”
```

in the source code, create a project file `chain.gpr` in the task directory, add the files `chain.c` (`chain.cpp`) and `meter.o` into this project, and then *compile* and/or *make* your program.

EXPERIMENTATION

You can experiment with the library by creating a text file `chain.in`. The file must contain two lines. The first line should contain the integer *N*, the number of atoms. The second line should contain a test case: a sequence of the atom labels, consisting of *N* different integers between 1 and *N*.

EXAMPLE FOR EXPERIMENTATION

`chain.in`	**`chain.out`**

`5`
`1 3 5 2 4`
	`Size=5`
	`Span(1,2)=3`
	`Span(1,3)=1`
	`Span(2,3)=2`
	`Span(4,5)=2`
	`Span(1,4)=4`
	`Span(3,5)=1`
	`Your Answer:`
	`1 3 5 2 4`
	`Max. Access/Atom:`
	`3`

CONSTRAINTS

- For the number of atoms, *N*, we have $5 \le N \le 10000$.
- Accessing any atom by `Span` more than four times will abort your program.
- Your program must not read or write any files.
- For the atom labels *x* and the atom positions *i*, we have $1 \le i, x \le N$.
- FreePascal library file names: `meter.ppw` and `meter.ow`.
- Pascal function declarations:

  ```
  function Size: integer;
  function Span(x, y: integer):integer;
  procedure Answer(i, x: integer);
  ```
- C/C++ library file names: `meter.h` and `meter`.o.

  ```
  int Size(void);
  int Span(int x, int y);
  void Answer(int i, int x);
  ```

GRADING

If your answer is correct and no atom was accessed more than *three* times then you obtain full score. If your answer is correct but some atom was accessed *four* times then you obtain 50% of the scores. If the answer is incorrect or any atom was accessed *five* times, then you receive 0 score.