Conqueror's battalion

Input file:	`none`	**100 points**
Output file:	`none`	**Time limit:** 1 s
Source Code:	`conquer.pas/.c/.cpp`	**Memory limit:** 16 MB

In the whole history of mankind one can find several curious battles, like the following one in France, in 1747...

There was a fortress in Bassignac-le-Haut, a small village lying on the left bank of river Dordogne, just over the Chastang dam. From the dam up to the fortress there was a wide staircase made out of red marble. One day in the morning, the guard spotted a large battalion approaching the fortress, with a dreaded leader – The Conqueror.

When The Conqueror reached the fortress, he was already awaited by its commandant. As the commandant had only a small part of his soldiery available, he proposed to The Conqueror: *"I see that you have many soldiers behind you, standing on the stairs. We can play a small 'game': In each round, you will divide your soldiers into two groups in an arbitrary way. Then I will decide which one of them stays and which one goes home. Each soldier that stays will then move up one stair. If at least one of your soldiers reaches the uppermost stair, you will be the winner, in the other case, you will be the loser. And your destination will be the dam down there... "*, added the commandant, pointing to the Chastang dam by his hand.

The Conqueror immediately liked this game, so he agreed and started to 'conquer'.

Task

Your role is The Conqueror's now. There are N stairs to the fortress ($2 \leq N \leq 2\,000$) and you have at most $1\,000\,000\,000$ soldiers. For each stair, you are given the number of soldiers standing on it, with number 1 being the uppermost stair and N the bottom one. None of your soldiers stands on stair 1 at the beginning.

For each starting position given to your program, if the position is winning (i.e. there is a strategy that enables you to win the game regardless of your opponent's moves), your program should win. Otherwise it should just play the game (and lose) correctly.

This is an interactive problem; you will play against a library as specified below. In each round, your program will describe a group of soldiers to our library. The library returns 1 or 2 specifying which group of soldiers should stay (1 means the group you described, 2 means the rest of the soldiers). In case the game ends (either because you won or there are no more soldiers in the game), the library will terminate your program correctly. Your program may not terminate in any other way.

Library interface

The library `libconq` provides two routines:

- `start` – returns the number N and fills an array *stairs* with numbers of soldiers standing on the stairs (i.e. there are $stairs[i]$ soldiers standing on stair i)

- `step` – takes an array *subset* (with at least N[1] elements), describing the group of soldiers you chose, and returns 1 or 2 as described above; the group of soldiers is specified by numbers of soldiers on each stair, as in the `start` function

If you fail to specify a valid group of soldiers, the game will be terminated and your program will score zero points for the particular test case. **Please note that also in C/C++ the stairs are numbered starting from 1.**

Following are the declarations of these routines in FreePascal and C/C++:

```
procedure start(var N: longint; var stairs:array of longint);
function step(subset:array of longint): longint;

void start(int *N, int *stairs);
int step(int *subset);
```

Below you can find examples of library usage in both FreePascal and C/C++; both fragments do the same – start the game and then play one round, with the chosen group containing all soldiers on randomly chosen stairs. Your real program will probably play the rounds in an infinite loop.

You are strongly encouraged to define the arrays `stairs` and `subset` in the same way as they are defined in the example below. (Note that the FreePascal library returns its answer in the first N elements of the array regardless of how you defined it, the C/C++ library returns its answer in the elements with indices 1 to N.)

FreePascal example:

```
uses libconq;
var stairs: array[1..2000] of longint;
    subset: array[1..2000] of longint;
    i,N,result: longint;

...
start(N,stairs);
...
for i:=1 to N do
  if random(2)=0 then subset[i]:=0
                 else subset[i]:=stairs[i];
result:=step(subset);
...
```

C/C++ example:

```
#include "libconq.h"
int stairs[2001];
int subset[2001];
int i,N,result;

...
start(&N, stairs);
...
for (i=1;i<=N;i++)
  if (rand()%2==0) subset[i]=0;
             else subset[i]=stairs[i];
result=step(subset);
...
```

You have to link the library to your program – by `uses libconq;` in FreePascal and by `#include "libconq.h"` in C/C++, where you have to compile your program by adding `libconq.c` to the compiler arguments.

[1] $N+1$ elements in C/C++, see below

An example of the game

You:	Library:
`start(N,stairs)`	N=8, stairs=(0,1,1,0,3,3,4,0)
`step((0,1,0,0,1,0,1,0))`	returns 2
`step((0,1,0,0,0,1,0,0))`	returns 2
`step((0,0,0,3,2,0,0,0))`	returns 1
`step((0,0,2,0,0,0,0,0))`	returns 2
`step((0,1,0,0,0,0,0,0))`	returns 2
`step((0,1,0,0,0,0,0,0))`	no return: you won

Resources

On the web page you may find example libraries for both C/C++ and FreePascal. These libraries are different from those that will be used during testing. You may use them to make sure your library calls are correct. The example library reads the input from the file `libconq.dat`, containing two lines. On the first line is the number N of stairs, the second line contains N integers – the numbers of soldiers on each of the stairs $1..N$.

The file `libconq.dat` for the example above would look like this:

```
8
0 1 1 0 3 3 4 0
```