INOI, Iranian National Olympiad in Informatics
Online Contest, April 2008
Practice Exam

Monsters Inc.

Time limit for each test: 400 miliseconds[1]
Memory limit: 44 megabytes[2]

They are n monsters working in Monsters Inc. An external enquirer organization wants to sort these monsters according to their salary but does not have access to the company's records. The only thing this organization can do, is to ask "whether the i^{th} monster is paid less than the j^{th}". You should help the enquirer organization to do the task.

We know that no two monsters have the same salary. We also know that the monsters are short tempered and if anyone asks them more than 2000000 questions, they get extremely mad, stop answering the questions, and ruin everything.

Problem

Write a program that

- asks the number of monsters from a *library*,
- uses the *library* to question the monsters on their salaries,
- reports a list of the monsters to the *library* sorted by their salaries in increasing order.

Library

The following functions are defined in the library of this problem:

`int init();`
This function returns the number of monsters, n ($1 \le n \le 10000$). It should be called exactly once, before any other library functions.

`int compare(int i, int j);`
This function asks the question

"Is the salary of the i^{th} monster less than the salary of the j^{th} monster?"

and returns the answer[3]. A return value of 1 indicates "yes" while a return value of 0 indicates "no".

`void report(int *index);`
You should call this function at the end to report the ordering of monsters to the library. This function never returns and calling it ends the execution of your program. The argument of this function is a pointer to an array of n elements of type `int` which indicates the monsters in increasing order of their salary.[4] The contents of this array should be a permutation of $\{1, \ldots, n\}$.

[1]Time limit is in fact 500 miliseconds, and the library will use at most 100 miliseconds.
[2]Memory limit is in fact 64 megabytes, and the library will use at most 20 megabytes.
[3]The monsters are numbered 1 to n.
[4]index[0] is the number of monster with the lowest salary and index[n-1] is the number of monster whose salary is the highest.

Your program should not read from the standard input or write to the standard output. It should not open any files and it should not try to access memory outside of its domain[5]. Breaking these rules might result in your disqualification.

You are provided two files named `sort.h` and `sortlib.cpp` for testing purposes. You should put this directive above your code: `#include "sort.h"`
For compiling your program, put these files next to your program and run this command:

```
g++ -O2 -static sort.cpp sortlib.cpp -lm
```

A file named `ssort.cpp` is given to you to serve as an example of using the provided library. Please, note that:

1. The provided `ssort.cpp` does not do anything interesting. It is only provided as an example to show how to use the library.
2. When we are testing your programs, they will be linked to a *different* library which implements the functions declared in `sort.h`. As long as you use the library in the prescribed manner, this should not have any implications for your program.

Testing

You can download the test library from the contest's interface. The provided test library reads n from the first line of standard input. It then reads n space-separated integers from the second line. The i^{th} number is the salary of the i^{th} monster.

When using the contest's "Test Run" feature, your program interacts with this same test library, so the file you provide as input should have the above format.

You are free to change `sortlib.cpp`, but it is advised not to change `sort.h`.

Grading

On any test case, if your program exceeds its time or memory limits, or if it does not properly interact with the test library, it will lose the mark of that test case.

Also, if your program makes more than 2000000 calls to the `compare` function, it will be stopped and it will not get any marks for the test case. Obviously, if the final ordering is wrong, your program will not get any marks. Even if you *do* provide the correct ordering, if the ordering is not uniquely determined by the asked questions, you will *not* get any points for that test case.

If your program does not fall into any of the above categories, the percentage of your grade on that test case is determined by the following formula,

$$\min\left(1, \frac{4n\lceil 1+\log_2(n)\rceil}{questions+1}\right) \times 100\%,$$

where *questions* denotes the number of times your program has called the `compare` function.

Interaction

The following is a simple interaction with the library.

Function call	Description
`int n=init();`	The value of n is set to the return value of this function, which is 3 in this example.
`int x=compare(1, 2);`	The value of x is set to the return value of this function, which is 1 in this example.
`int y=compare(1, 3);`	The value of y is set to the return value of this function, which is 1 in this example.
`int z=compare(2, 3);`	The value of z is set to the return value of this function, which is 0 in this example.
`int a[3];`	An array is constructed to give the final answer.
`a[0]=1;a[1]=3;a[2]=2;`	`a` is filled.
`report(a);`	The answer is sent to the library.

[5]Particularly, you should not try to access the library's memory.

The following is an example input file for the testing library.

```
10
1 3 7 2 4 9 6 5 8 0
```