Task: REC
Rectangle Game

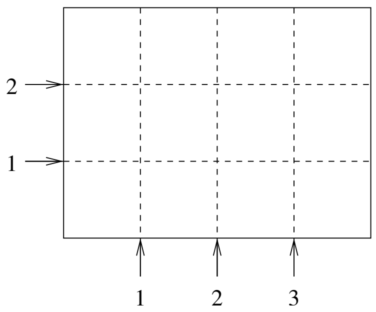
Official English Version

Nowy Sącz, POLAND 2005
The 17-th International
Olympiad in Informatics

Day 2. Source file `rec.*` *Monday, 22–08–2005*

Available memory: 32 MB. Maximum running time: 14 s*.

We consider a two-player game. The players are given an $x \times y$ rectangle (where x and y are positive integers). The players take turns moving. A move consists of dividing a rectangle into two rectangles with a single vertical or horizontal cut. The resulting rectangles must have positive integer dimensions.



Possible cuts of a 4×3 rectangle.

After each cut, the smaller rectangle (that is the one with smaller area) is discarded and the other one is passed to the other player. If the rectangle is cut into two equal halves, then one half is discarded. The player who receives a 1×1 rectangle, and therefore is not able to make a move, loses the game.

Your task is to write a program to play and win the rectangle game. The program must use a special library to play the game. The library provides you with functions `dimension_x()` and `dimension_y()` that return the dimensions of the rectangle. Initial dimensions of the rectangle are integers from 1 to 100 000 000. At least one dimension is greater than 1. Moreover, in 50% of test cases the dimensions do not exceed 25.

There is also a procedure `cut(dir, position)`, that is called by your program to make moves. Parameters `dir` and `position` describe the direction and the position of a cut respectively. The parameter `dir` must be one of the two values: `vertical` or `horizontal`. If $\texttt{dir} = \texttt{vertical}$ then the cut is vertical, and the parameter `position` specifies the x-coordinate of the cut (see the figure above) and you must ensure that $1 \leq \texttt{position} \leq \texttt{dimension\_x}() - 1$. If $\texttt{dir} = \texttt{horizontal}$, then the cut is horizontal and the parameter `position` specifies the y-coordinate of the cut and you must ensure that $1 \leq \texttt{position} \leq \texttt{dimension\_y}() - 1$.

When your program is started, it will act as one player for one game. Your program plays first — it must cut the initial rectangle. When your program calls the `cut` procedure, your move is recorded and control is passed to your program's opponent. After the opponent moves, control returns to your program. Values returned by `dimension_x()` and `dimension_y()` will reflect the result of your move and your opponent's move. As soon as your program wins, loses or makes an illegal move (i.e. calls the `cut` procedure with invalid parameters) it will be terminated. Termination of your program is an automatic process, so your program should keep making moves as long as possible. You can assume that for the test data, there always exists a winning strategy for your program.

Your program must not read or write any files, it must not use standard input/output, and it must not try to modify any memory outside your program. Violating any of these rules may result in disqualification.

* You can assume that during grading library overhead will not exceed 4 s.

Experimentation

To let you experiment with the library, you are given example opponent libraries: their sources are in `preclib.pas`, `creclib.c` and `creclib.h` files. The library can be downloaded from `http://contest/`. They implement a very simple strategy. When you run your program, it will be playing against these simple players. Feel free to modify them, and test your program against a better opponent. However, during the evaluation, your program will be playing against a different opponent.

When you submit your program using the TEST interface it will be compiled with the unmodified example opponent library. The submitted input file will be given to your program standard input. The input file should consist of two lines, each containing one integer. The first line should contain the initial width, and the second line should contain the initial height of the rectangle. These dimensions are read by the example opponent library.

If you modify the implementation part of the `preclib.pas` library, please recompile it using the following command: `ppc386 -O2 preclib.pas`. This command produces files `preclib.o` and `preclib.ppu`. These files are needed to compile your program, and should be placed in the directory, where your program is located. Please do not modify the interface part of the `preclib.pas` library.

If you modify the `creclib.c` library, please remember to place it (together with `creclib.h`) in the directory, where your program is located — they are needed to compile it. Please do not modify the `creclib.h` file.

You are also provided with two simple programs illustrating usage of the above libraries: `crec.c` and `prec.pas`. (Please remember, that these programs are not correct solutions.) You can compile them using the following commands:

```
gcc -O2 -static crec.c creclib.c -lm
g++ -O2 -static crec.c creclib.c -lm
ppc386 -O2 -XS prec.pas
```

Library

You are given a library providing the following functionality:

- **FreePascal Library** (`preclib.ppu`, `preclib.o`)

  ```
  type direction = (vertical, horizontal);
  function dimension_x(): longint;
  function dimension_y(): longint;
  procedure cut(dir: direction; position: longint);
  ```

  Include the following statement in your source file `rec.pas`:
  `uses preclib;`
  To compile your program, copy the files `preclib.o` and `preclib.ppu` to the directory, where your source file is placed and run the following command:
  `ppc386 -O2 -XS rec.pas`

  File `prec.pas` gives an example of how to use the `preclib` library.

- **GNU C/C++ Library** (`creclib.h`, `creclib.c`)

  ```
  typedef enum __direction {vertical, horizontal} direction;
  int dimension_x();
  int dimension_y();
  void cut(direction dir, int position);
  ```

  Include the following statement in your source file (`rec.c` or `rec.cpp`):
  `#include "creclib.h"`
  To compile your program, copy the files `creclib.c` and `creclib.h` to the directory, where your source file is placed and run the following command:
  `gcc -O2 -static rec.c creclib.c -lm`
  or:
  `g++ -O2 -static rec.cpp creclib.c -lm`
  The file `crec.c` gives an example of how to use the library in C.

Sample interaction

Below there is a sample interaction between your program and the judging library. It shows how a sample game can proceed. The game starts with a 4×3 board. There exists a winning strategy for this position.

Your program calls	**What happens**
`dimension_x()`	returns 4
`dimension_y()`	returns 3
`cut(vertical, 1)`	your cut is recorded and a 3×3 board is passed to your opponent, who cuts it to a 3×2 board; after this, control is passed back to your program
`dimension_x()`	returns 3
`dimension_y()`	returns 2
`cut(horizontal, 1)`	your cut is recorded and a 3×1 board is passed to your opponent, who cuts it to a 2×1 board; after this, control is passed back to your program
`dimension_x()`	returns 2
`dimension_y()`	returns 1
`cut(vertical, 1)`	your cut gives a 1×1 board, so you win; your program is terminated automatically