Task: KUC
Cook

Olimpiada Informatyczna

XXIV OI, Stage III, Day two. Source file `kuc.*` Available memory: 8 MB. *12.04.2017*

Byteasar, who is working as a cook in a restaurant, has n orders to fill. Each order is written on a slip of paper, and all slips are speared on a spindle. Byteasar's goal is to fill all orders as quickly as possible. There are quite a few, so for efficiency, he will only take the orders from the top of the spindle. He may however work on a few dishes at once. Specifically, if there are k slips remaining on the spindle, he may:

- take the top slip and fill this order in time `jedno`(k).
- If $k > 1$, he may take two slips from the top and fill those orders in time `dwa`(k).
- If $k > 1$, he may take $\lfloor k/2 \rfloor$ orders from the top and fill those in time `polowa`(k).

These operations take a toll on Byteasar, who has only a certain amount of energy, initially equal to e. Namely, the third operation is so demanding that it decrements Byteasar's energy by 1, the middle one is energy-neutral, whereas working on just one order is so pleasant and relaxing that the first operation increments Byteasar's energy by 1. Byteasar's energy may never drop below 0, but other than he intends to do whatever it takes so that the time to serve all orders is minimized, i.e., his final energy, as long as non-negative, does not matter.

Write a program that will determine the minimum time sufficient to fill all orders. Your program is to communicate with a library that supplies all the necessary information on Byteasar's and his kitchen's state. You are well advised to take heed of the memory limit in this task.

Communication

To use the library, write the following at the beginning of your program:

- **C/C++**: `#include "ckuclib.h"`
- **Pascal**: `uses pkuclib;`

The library supplies the following functions and procedures:

- `dajn`, `daje`
  The first function returns an integer n, which is the number of orders to be filled, whereas the second – an integer e, which is Byteasar's initial energy.
  - **C/C++:**
    ```
    int dajn();
    int daje();
    ```
  - **Pascal:**
    ```
    function dajn: LongInt;
    function daje: LongInt;
    ```
- `jedno`(k), `dwa`(k), `polowa`(k)
  These functions return the time to execute the actions of filling one, two, and half of outstanding orders ($\lfloor k/2 \rfloor$ to be exact) respectively when there are k slips on the spindle. For $k = 1$, the values returned by `dwa`(k) and `polowa`(k) should be ignored. The reported time is an integer in the range from 1 to 10^7.
  - **C/C++:**
    ```
    int jedno(int k);
    int dwa(int k);
    int polowa(int k);
    ```
  - **Pascal:**
    ```
    function jedno(k: LongInt): LongInt;
    function dwa(k: LongInt): LongInt;
    function polowa(k: LongInt): LongInt;
    ```
- `odpowiedz`$(wynik)$
  Reports to the library that *wynik* is the minimum time sufficient for filling all the orders. Calling this function **terminates the execution of your program**.
  - **C/C++:** `void odpowiedz(int wynik);`
  - **Pascal:** `procedure odpowiedz(wynik: LongInt);`

Your program **cannot** read any data (neither from standard input, nor from any file). Moreover, it **cannot** write anything to files nor standard output. It may write to the standard diagnostic output (`stderr`) – keep in mind though that this takes precious time. The library can be queried multiple times for the same values.

Grading

The set of tests consists of the following subsets. Within each subset, there may be several test groups.

Subset	Property	Score
1	$n, e \leq 1000$	12
2	$n, e \leq 50\,000$	8
3	$n, e \leq 1\,000\,000$	80

Sample execution of a program

C/C++	Pascal	Wynik
`n = dajn();`	`n := dajn;`	3
`e = daje();`	`e := daje;`	1
`p[3] = polowa(3);`	`p[3] := polowa(3);`	1
`p[2] = polowa(2);`	`p[2] := polowa(2);`	4
`j[2] = jedno(2);`	`j[2] := jedno(2);`	2
`j[1] = jedno(1);`	`j[1] := jedno(1);`	5
`d[2] = dwa(2);`	`d[2] := dwa(2);`	6
`odpowiedz(7);`	`odpowiedz(7);`	—

The above execution is formally correct, but may report an incorrect result. From the answers it can be inferred that performing `polowa` followed by `dwa` (at a cost of $p[3] + d[2] = 7$) is better than performing a single `polowa` followed by two `jedno` (at a cost of $p[3] + j[2] + j[1] = 8$). Moreover, performing `jedno` is pointless when there are three outstanding orders, as $p[3] = 1$ and $j[3] \geq 1$. Performing `polowa` twice is not possible, since $e = 1$. But without knowing the cost of `dwa(3)`, we cannot tell if performing `dwa` followed by `jedno` would cost less than 7.

Experiments

In the `dlazaw` folder, there is a sample library that may be used to test the formal correctness of your program. This library reads the information from the standard input, formatted as follows:

- in the first line, two integers n and e;
- in the second line, n integers from the range $[1, 10^7]$ – the values of the function `jedno` for successive $k = 1, \ldots, n$;
- in third and fourth line respectively, the values of the functions `dwa` and `polowa` (using same format); the values for $k = 1$ have to be provided, even though they will be ignored.

A sample input for the library is provided in a file `kuc0.in`. Once the procedure `odpowiedz` is executed, the library writes the reported answer to the standard output.

The same directory contains sample solutions (programs) `kuc.c`, `kuc.cpp`, and `kuc.pas` that use the library. These are incorrect; in particular, they always repeatedly perform the operation `polowa` until there is only one slip left, at which point they perform the operation `jedno`.

To compile your program together with the library, execute the following command:

- **C**: `gcc -O2 -static ckuclib.c kuc.c -lm -std=gnu99`
- **C++**: `g++ -O2 -static ckuclib.c kuc.cpp -lm -std=c++11`
- **Pascal**: `ppc386 -O2 -XS -Xt kuc.pas`

Your program file and the library file should reside in the same folder.