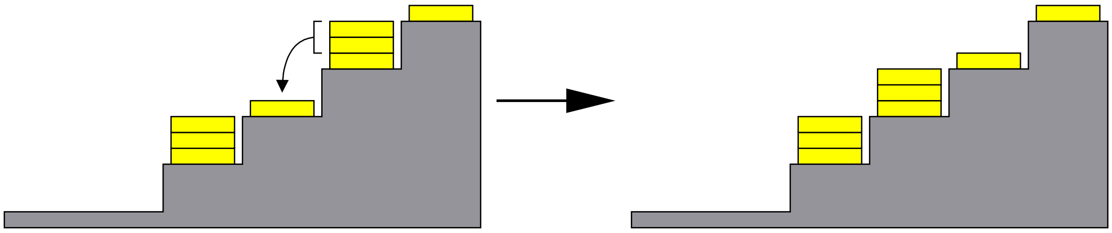
Zadanie: STAIRS
Schody

dzień 4, plik źródłowy `stairs.*`, dostępna pamięć 32 MB *CPSPC 2004, 11 czerwca*

Kiedy studenci FMFI UK z Bratysławy są zmuszeni do czekania na schodach, zazwyczaj spędzają go grają w następującą grę dwuosobową. Na początku umieszczają na schodach wszystkie monety, które mają. Następnie wykonują posunięcia na przemian. Posunięcie polega na wybraniu jednego ze schodków i przełożeniu pewnej liczby monet z tego schodka na schodek znajdujący się bezpośrednio poniżej. Gracz, który położy ostatnią monetę na ziemi pod schodami, wygrywa i zbiera wszystkie monety.



Przykładowe posunięcie — student przełożył 2 monety z trzeciego schodka.

Zadanie

Twoim zadaniem jest napisać program grający w tę grę z przeciwnikiem zadanym za pomocą modułu grającego.

Modułu grający

Komunikacja z modułem grającym odbywa się za pomocą dwóch funkcji.

Pascal:
Na początku programu należy dodać linię `uses uStairs;`.

```
type TARRAY=array[0..1000] of longint;
procedure Init_game(var n:longint; var stairs:TARRAY);
procedure Turn(from,coins:longint; var hisfrom,hiscoins:longint);
```

C:
Na początku programu należy dodać linię `#include "uStairs.h"`.

```
void Init_game(int *n, int *stairs);
void Turn(int from, int coins, int *hisfrom, int *hiscoins);
```

Zalecane jest kompilowanie programów za pomocą dostarczonego skryptu `compile`, który potrafi odpowiednio zlinkować program.

Moduł oczekuje, że funkcja `Init_game` zostanie wywołana dokładnie raz na początku gry. Funkcja ta zwraca liczbę schodków w grze (zmienna `n`) oraz liczbę monet znajdujących się na każdym ze schodków (tablica `stairs`). Liczba schodków nie większa niż 1 000.

Ruch wykonuje się wywołując funkcję `Turn`. W parametrach `from` i `coins` należy podać numer schodka, z którego chcesz zabrać monety oraz liczbę monet, które przekładasz na niższy schodek. Na zmiennych `hisfrom` i `hiscoins` moduł grający zwraca swój ruch.

Nie powinieneś kończyć programu. Moduł zrobi to za Ciebie kiedy zostanie wyłoniony zwycięzca rozgrywki.

Twój program nie powinien czytać, ani pisać. Powinien tylko grać w grę używając modułu grającego. Możesz założyć, że podczas sprawdzania programu, moduł grający będzie grał optymalnie. W sytuacjach, w których można wygrać Twój program powinien wygrać, natomiast jeśli sytuacja jest przegrana, to Twój program powinien wykonywać do końca rozgrywki jakiekolwiek poprawne ruchy.

Przykład

Przykładowa rozgrywka dla `n` $= 4$ i `stairs` $= (47, 2, 1, 0, 1)$:

Gracz	Posunięcie	Sytuacja po wykonaniu posunięcia
1	1 moneta ze schodka 4	47 2 1 1 0
2	1 moneta ze schodka 2	47 3 0 1 0
1	1 moneta ze schodka 3	47 3 1 0 0
2	2 monety ze schodka 1	49 1 1 0 0
1	1 moneta ze schodka 1	50 0 1 0 0
2	1 moneta ze schodka 2	50 1 0 0 0
1	1 moneta ze schodka 1	51 0 0 0 0

Gracz 1 wygrał.