Zadanie: XMAS
Choinka

Etap CPSPC 2007. Dzień drugi. Plik źródłowy `xmas.*` *12.06.2007*

Dostępna pamięć: 64 MB.

Wszedłeś w posiadanie ślicznej, nowiutkiej choinki (no cóż, może to i nie czas na bożonarodzeniowe zakupy, ale przecena była) i pudełka pełnego choinkowych ozdób. Bombki w pudełku są czterech rodzajów.

Chinka składa się z punktów, w których można umieszczać ozdoby. Niektóre z tych punktów są połączone gałązkami. Drzewko jest spójne: wszystkie punkty są połączone (niekoniecznie bezpośrednio) gałązkami i w połączeniach nie występują cykle.

Założyłeś się z kumplem, że nie będzie on w stanie powstrzymać Cię od ozdobienia choinki w wypasiony sposób. (Nie mów, że nie znasz zasad wypasionego ozdabiania choinek! Drzewko jest wypasione, jeśli żadna gałązka nie łączy dwóch punktów ozdobionych bombką tego samego rodzaju.)

Razem z kumplem będziecie na przemian przyozdabiać choinkę. W swoim ruchu (Ty zaczynasz) każdy z Was może ozdobić nieozdobiony punkt, przy czym nie można złamać zasady wypasionego ozdabiania choinki.

Jeśli uda Ci się doprowadzić do sytuacji, w której wszystkie punkty choinki będą przyozdobione, wygrałeś. Domyślasz się, że w przyciwnym wypadku wygrał Twój kumpel.

Biblioteka

To zadanie nie posiada wejścia ani wyjścia. Wszelka komunikacja będzie odbywała się poprzez specjalną bibliotekę, która pozwoli Ci poznać strukturę drzewka i zasymuluje grę Twojego kumpla (dalej zwanego przeciwnikiem).

Biblioteka nazywa się `tree_lib` i powinieneś ją dołączyć (ang. *link*) pisząc `#include "tree_lib.h"` albo `use tree_lib;` (w zależności od wybranego języka). Biblioteka dostarcza następujące funkcje:

- ```
  int init();
  function init: integer;
  ```

  Wywołaj tę funkcję dokładnie raz przed wywołaniem jakiejkolwiek innej funkcji z biblioteki. Zwróci ona N ($1 \leq N \leq 1\,000\,000$) — liczbę ozdabialnych punktów. Punkty będą numerowane liczbami całkowitymi od 1 do N.

- ```
  int neighbor_count( int point );
  function neighbor_count( point: integer ): integer;
  ```

  Ta funkcja zwraca liczbę punktów bezpośrednio połączonych z danym punktem. Jeśli parametr *point* jest poza zakresem $1, \ldots, N$, zwracane jest 0.

- ```
  int neighbor( int point, int index );
  function neighbor( point, index: integer ): integer;
  ```

  Funkcja zwraca sąsiada punktu *point* o numerze *index* (*index* liczony jest od 1). Jeśli którykolwiek z parametrów jest poza zakresem, zwracane jest 0.

- ```
  char decoration( int point );
  function decoration( point: integer ): char;
  ```

Ta funkcja zwraca rodzaj bombki wiszącej w punkcie *point*. Jest to jedna z wartości ‘A’, ‘B’, ‘D’ lub ‘S’. Dla nieozdobionego lub nieistniejącego punktu funkcja zwraca ‘ ’.

- `int enemy();`
  `function enemy(): integer;`

  Ta funkcja może być wywołana tylko wtedy, kiedy nadszedł czas na ruch przeciwnika. Do Ciebie należy upomnienie się o wykonanie tego ruchu. Funkcja zwraca numer punktu, który ozdobił przeciwnik.

- `void decorate( int point, char decoration );`
  `procedure decorate( point: integer; decoration: char );`

  Użyj tej procedury aby ozdobić punkt *point* bombką rodzaju *decoration* (jedno z ‘A’, ‘B’, ‘D’, ‘S’). Możesz ją wywołać tylko wtedy, kiedy jest pora na Twój ruch.

Jeśli ozdobisz nieistniejący punkt, spróbujesz powtórnie ozdobić ozdobiony punkt, umieścisz bombkę niezgodnie z zasadą wypasionego ozdabiania choinki lub olejesz kolejność ruchów, Twój program zostanie zakończony i przegrasz. Kiedy całe drzewko zostanie ozdobione, Twój program również zostanie zakończony, ale wygrasz.