Iteradors que s'enganxen a elements — X66581_ca

Típicament, executar `++` sobre un iterador que es troba al end de la llista produeix error d'execució, i executar `--` sobre un iterador que es troba al begin de la llista també produeix error d'execució. Per començar, en aquest exercici modificarem la subclasse `iterator` de la classe List de manera que els errors d'execució abans esmentats ja no es produiran. Simplement, en tals casos els iteradors no es mouran.
Després modificarem la classe `iterator` afegint tres nous mètodes `hook`, `stopHook`, `hasActiveHook`, i canviant el comportament dels mètodes `++` i `--` com descrivim a continuació.
Suposem que un cert iterador `it` apunta a un cert element e_1 d'una llista. Llavors, una crida `it.hook()` provocarà que, a partir d'ara, `it` quedi enganxat a aquest element, i se l'endugui amb ell a dreta o esquerra quan rep crides `++` o `--`, respectivament.
Per a ser més precisos, suposem que a la dreta de e_1 hi ha un cert element e_2 (és a dir, e_2 està una unitat més a prop de l'`end` de la llista que e_1. Llavors, una crida `it++` o `++it`, enlloc de fer que `it` apunti a e_2, el que provocarà és que e_1 i e_2 intercanviin les seves posicions, i `it` seguirà apuntant a e_1.
En el cas particular que e_1 ja no tingui ningú a la dreta (i per tant e_1 sigui l'últim element de la llista, i a la seva dreta hi hagi l'`end` de la llista), llavors una crida `it++` o `++it` no provocarà cap canvi.
Anàlogament, suposem que a l'esquerra de e_1 hi ha un cert element e_2' (és a dir, e_2' està una unitat més a prop del `begin` de la llista que e_1. Llavors, una crida `it--` o `--it`, enlloc de fer que `it` apunti a e_2', el que provocarà és que e_1 i e_2' intercanviin les seves posicions, i `it` seguirà apuntant a e_1.
En el cas particular que e_1 ja no tingui ningú a l'esquerra (i per tant e_1 sigui justament el `begin` de la llista), llavors una crida `it--` o `--it` no provocarà cap canvi.
Una crida posterior `it.stopHook()` cancel.la aquest comportament alternatiu de `it`, i torna al comportament usual, de manera que, a partir de llavors, les crides `++` el mouen a la dreta i les crides `--` el mouen a l'esquerra, sense provocar cap intercanvi entre posicions d'elements de la llista.
Una crida `it.hasActiveHook()` retorna cert si `it` té un hook actiu, és a dir, si en algun moment hi ha hagut una crida del tipus `it.hook()`, i després de l'última d'aquesta mena de crides no hi ha hagut cap crida del tipus `it.stopHook()`.
Fixeu-vos en aquest exemple per tal d'acabar d'entendre-ho:

```
List<int> l0, l1;
List<int>::iterator a, b, c, d;

l0.push_back(1);     // l0: 1,
l0.push_back(2);     // l0: 1,2,
l0.push_back(3);     // l0: 1,2,3,
l1.push_back(4);     // l1: 4,
l1.push_back(5);     // l1: 4,5,
l1.push_back(6);     // l1: 4,5,6,

a = l0.begin();      // l0: 1a,2,3,
b = l0.end();        // l0: 1a,2,3,b
c = l1.begin();      // l1: 4c,5,6,
```

```
d = l1.end();          // l1: 4c,5,6,d

a--;                   // l0: 1a,2,3,b
a++;                   // l0: 1,2a,3,b
b++;                   // l0: 1,2a,3,b
b--;                   // l0: 1,2a,3b,
a.hook();              // l0: 1,2[a],3b,
a--;                   // l0: 2[a],1,3b,
a--;                   // l0: 2[a],1,3b,
a++;                   // l0: 1,2[a],3b,
a++;                   // l0: 1,3b,2[a],
a++;                   // l0: 1,3b,2[a],
a--;                   // l0: 1,2[a],3b,
b--;                   // l0: 1,2[a]b,3,
a++;                   // l0: 1,3,2[a]b,
a++;                   // l0: 1,3,2[a]b,
a--;                   // l0: 1,2[a]b,3,
a--;                   // l0: 2[a]b,1,3,
a--;                   // l0: 2[a]b,1,3,
b--;                   // l0: 2[a]b,1,3,
b++;                   // l0: 2[a],1b,3,
b.hook;                // l0: 2[a],1[b],3,
b--;                   // l0: 1[b],2[a],3,
a--;                   // l0: 2[a],1[b],3,
b++;                   // l0: 2[a],3,1[b],
a.stopHook();          // l0: 2a,3,1[b],
a--;                   // l0: 2a,3,1[b],
a++;                   // l0: 2,3a,1[b],
a++;                   // l0: 2,3,1a[b],
a++;                   // l0: 2,3,1[b],a
c.hook();              // l1: 4[c],5,6,d
c++;                   // l1: 5,4[c],6,d
c++;                   // l1: 5,6,4[c],d
c++;                   // l1: 5,6,4[c],d
d--;                   // l1: 5,6,4[c]d,
d.hook();              // l1: 5,6,4[c][d],
d--;                   // l1: 5,4[c][d],6,
c--;                   // l1: 4[c][d],5,6,
d--;                   // l1: 4[c][d],5,6,
c--;                   // l1: 4[c][d],5,6,
d++;                   // l1: 5,4[c][d],6,
c.stopHook;            // l1: 5,4c[d],6,
d++;                   // l1: 5,6,4c[d],
c++;                   // l1: 5,6,4[d],c
c++;                   // l1: 5,6,4[d],c
d++;                   // l1: 5,6,4[d],c
```

D'entre els fitxers que s'adjunten en aquest exercici, trobareu `list.hh`, a on hi ha una implementació de la classe genèrica `List`. Haureu d'implementar els tres nous mètodes `hook`, `stopHook` i `hasActiveHook` dins `list.hh` a la part pública de la classe `iterator`

(podeu trobar les capçaleres comentades dins `list.hh`), i modificar els dos mètodes `++` i els dos mètodes `--` convenientment (en realitat només cal modificar el pre-increment i el pre-decrement perquè el post-increment i post-decrement criden als primers). Necessitareu també algun atribut addicional per tal de recordar si l'iterador té un `hook` actiu, amb les convenients inicialitzacions.
Més concretament, heu de fer els canvis que s'indiquen en algunes parts del codi de list.hh:

```
  // Iterators mutables
  class iterator {
    friend class List;
    private:
      List *plist;
      Item *pitem;
      // Add new attributes to remember if the iterator has an active 'hook'

    public:

    iterator() {
      // Add initialization of new attributes.
    }


    // Adapt this function so that moving beyond boundaries does not trigger er
    // but leaves the iterator unchanged instead.
    // Also, add the necessary adaptations so that, when the iterator has an ac
    // instead of making the iterator point to next element (towards the end of
    // the iterator keeps pointing to the same element, and this element swaps
    // with the next one (towards the end of the list). In the event that there
    // a next element, nothing changes.
    // Preincrement
    iterator operator++()
    /* Pre: el p.i apunta a un element E de la llista,
       que no és el end() */
    /* Post: el p.i apunta a l'element següent a E
       el resultat és el p.i. */
    {
      if (pitem == &(plist->itemsup)) {
        cerr << "Error: ++iterator at the end of list" << endl;
        exit(1);
      }
      pitem = pitem->next;
      return *this;
    }

...

    // Adapt this function so that moving beyond boundaries does not trigger er
    // but leaves the iterator unchanged instead.
    // Also, add the necessary adaptations so that, when the iterator has an ac
    // instead of making the iterator point to previous element (towards the be
```

```
    // the iterator keeps pointing to the same element, and this element swaps
    // with previous one (towards the begin of the list). In the event that the
    // a previous element, nothing changes.
    // Predecrement
    iterator operator--()
    /* Pre: el p.i apunta a un element E de la llista que
       no és el begin() */
    /* Post: el p.i apunta a l'element anterior a E,
       el resultat és el p.i. */
    {
      if (pitem == plist->iteminf.next) {
       cerr << "Error: --iterator at the beginning of list" << endl;
       exit(1);
      }
      pitem = pitem->prev;
      return *this;
    }

...

    // Pre: Iterator 'this' (the implicit parameter) does not have an active ho
    //      and it points to an element of a list.
    //      In particular, 'this' does not point to the end of a list.
    // Post: 'it' keeps pointing to the same element and has an active hook to
    // Remove comment marks and implement this function:
    // void hook() {
    // }

    // Pre: 'this' has an active hook.
    // Post: 'this' does not have an active hook.
    // Remove comment marks and implement this function:
    // void stopHook() {
    // }

    // Pre:
    // Post: Returns true iff 'this' has an active hook.
    // Remove comment marks and implement this function:
    // bool hasActiveHook() const {
    // }

...
```

No cal decidir que passa amb assignacions entre iteradors existents, doncs no es consideraran en els jocs de proves.
D'entre els fitxers que s'adjunten a l'exercici també hi ha `main.cc` (programa principal), i el podeu compilar directament, doncs inclou `list.hh`. Només cal que pugeu `list.hh` al jutge.

Entrada

L'entrada del programa comença amb una declaració d'unes quantes llistes (`l0, l1, ...`) i uns quants iteradors (`a,b,c,...`), i després té una seqüència de comandes sobre les llistes i els iteradors declarats. Com que ja us oferim el `main.cc`, no cal que us preocupeu d'implementar la lectura d'aquestes entrades. Només cal que implementeu la extensió de la classe `iterator` abans esmentada.
Per simplificar, no hi haurà comandes que eliminin elements de les llistes, com `pop_back, pop_front i erase`. Podeu suposar que les comandes no fan coses extranyes, com fer `hook` d'un iterador que no apunta a cap element, ni `hasActiveHook` d'un iterador que no apunta a enlloc, i que sempre que un iterador sigui mogut, aquest estarà apuntant a alguna posició d'alguna llista (amb un element o l'end). Podeu suposar que les comandes faran `hook` sobre iteradors sense cap hook actiu, i que faran `stopHook` sobre iteradors que tinguin un hook actiu.

Sortida

Per a cada comanda d'escriptura sobre la sortida s'escriurà el resultat corresponent. El `main.cc` que us oferim ja fa això. Només cal que implementeu la extensió de la classe `iterator` abans esmentada.

Exemple d'entrada 1

```
List<int> l0 , l1 ;
List<int>::iterator a , b , c , d ;

l0 .push_back( 1 );     // l0: 1,
l0 .push_back( 2 );     // l0: 1,2,
l0 .push_back( 3 );     // l0: 1,2,3,
l1 .push_back( 4 );     // l1: 4,
l1 .push_back( 5 );     // l1: 4,5,
l1 .push_back( 6 );     // l1: 4,5,6,

a = l0 .begin();      // l0: 1a,2,3,
b = l0 .end();        // l0: 1a,2,3,b
c = l1 .begin();      // l1: 4c,5,6,
d = l1 .end();        // l1: 4c,5,6,d

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                 // l0: 1a,2,3,b

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                 // l0: 1,2a,3,b

cout<< l0 <<endl;
cout<< l1 <<endl;

b ++;                 // l0: 1,2a,3,b

cout<< l0 <<endl;
cout<< l1 <<endl;

b --;                 // l0: 1,2a,3b,

cout<< l0 <<endl;
cout<< l1 <<endl;

a .hook();            // l0: 1,2[a],3b,

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                 // l0: 2[a],1,3b,

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                 // l0: 2[a],1,3b,

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                 // l0: 1,2[a],3b,

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                 // l0: 1,3b,2[a],

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                 // l0: 1,3b,2[a],

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                 // l0: 1,2[a],3b,
```

```
cout<< l0 <<endl;
cout<< l1 <<endl;

b --;                  // l0: 1,2[a]b,3,

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                  // l0: 1,3,2[a]b,

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                  // l0: 1,3,2[a]b,

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                  // l0: 1,2[a]b,3,

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                  // l0: 2[a]b,1,3,

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                  // l0: 2[a]b,1,3,

cout<< l0 <<endl;
cout<< l1 <<endl;

b --;                  // l0: 2[a]b,1,3,

cout<< l0 <<endl;
cout<< l1 <<endl;

b ++;                  // l0: 2[a],1b,3,

cout<< l0 <<endl;
cout<< l1 <<endl;

b .hook();               // l0: 2[a],1[b],3,

cout<< l0 <<endl;
cout<< l1 <<endl;

b --;                  // l0: 1[b],2[a],3,

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                  // l0: 2[a],1[b],3,

cout<< l0 <<endl;
cout<< l1 <<endl;

b ++;                  // l0: 2[a],3,1[b],

cout<< l0 <<endl;
```

```
cout<< l1 <<endl;

a .stopHook();         // l0: 2a,3,1[b],

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                  // l0: 2a,3,1[b],

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                  // l0: 2,3a,1[b],

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                  // l0: 2,3,1a[b],

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                  // l0: 2,3,1[b],a

cout<< l0 <<endl;
cout<< l1 <<endl;

c .hook();             // l1: 4[c],5,6,d

cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                  // l1: 5,4[c],6,d

cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                  // l1: 5,6,4[c],d

cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                  // l1: 5,6,4[c],d

cout<< l0 <<endl;
cout<< l1 <<endl;

d --;                  // l1: 5,6,4[c]d,

cout<< l0 <<endl;
cout<< l1 <<endl;

d .hook();             // l1: 5,6,4[c][d],

cout<< l0 <<endl;
cout<< l1 <<endl;

d --;                  // l1: 5,4[c][d],6,

cout<< l0 <<endl;
cout<< l1 <<endl;
```

```
c --;                  // l1: 4[c][d],5,6,

cout<< l0 <<endl;
cout<< l1 <<endl;

d --;                  // l1: 4[c][d],5,6,

cout<< l0 <<endl;
cout<< l1 <<endl;

c --;                  // l1: 4[c][d],5,6,

cout<< l0 <<endl;
cout<< l1 <<endl;

d ++;                  // l1: 5,4[c][d],6,

cout<< l0 <<endl;
cout<< l1 <<endl;

c .stopHook();           // l1: 5,4c[d],6,

cout<< l0 <<endl;
cout<< l1 <<endl;

d ++;                  // l1: 5,6,4c[d],

cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                  // l1: 5,6,4[d],c

cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                  // l1: 5,6,4[d],c

cout<< l0 <<endl;
cout<< l1 <<endl;

d ++;                  // l1: 5,6,4[d],c

cout<< l0 <<endl;
cout<< l1 <<endl;
```

Exemple de sortida 1

```
1a,2,3,b
4c,5,6,d
1a,2,3,b
4c,5,6,d
1,2a,3,b
4c,5,6,d
1,2a,3,b
4c,5,6,d
1,2a,3b,
4c,5,6,d
1,2[a],3b,
4c,5,6,d
2[a],1,3b,
4c,5,6,d
2[a],1,3b,
4c,5,6,d
1,2[a],3b,
4c,5,6,d
1,3b,2[a],
4c,5,6,d
1,3b,2[a],
4c,5,6,d
1,2[a],3b,
4c,5,6,d
1,2[a]b,3,
4c,5,6,d
1,3,2[a]b,
4c,5,6,d
1,3,2[a]b,
4c,5,6,d
1,2[a]b,3,
4c,5,6,d
2[a]b,1,3,
4c,5,6,d
2[a]b,1,3,
4c,5,6,d
2[a]b,1,3,
4c,5,6,d
2[a],1b,3,
4c,5,6,d
2[a],1[b],3,
4c,5,6,d
1[b],2[a],3,
4c,5,6,d
2[a],1[b],3,
4c,5,6,d
2[a],3,1[b],
4c,5,6,d
2a,3,1[b],
4c,5,6,d
2a,3,1[b],
4c,5,6,d
2,3a,1[b],
4c,5,6,d
2,3,1a[b],
4c,5,6,d
2,3,1[b],a
4c,5,6,d
2,3,1[b],a
```

```
4[c],5,6,d
2,3,1[b],a
5,4[c],6,d
2,3,1[b],a
5,6,4[c],d
2,3,1[b],a
5,6,4[c],d
2,3,1[b],a
5,6,4[c]d,
2,3,1[b],a
5,6,4[c][d],
2,3,1[b],a
5,4[c][d],6,
2,3,1[b],a
4[c][d],5,6,
2,3,1[b],a
```

```
4[c][d],5,6,
2,3,1[b],a
4[c][d],5,6,
2,3,1[b],a
5,4[c][d],6,
2,3,1[b],a
5,4c[d],6,
2,3,1[b],a
5,6,4c[d],
2,3,1[b],a
5,6,4[d],c
2,3,1[b],a
5,6,4[d],c
2,3,1[b],a
5,6,4[d],c
```

Exemple d'entrada 2

```
List<int> l0 , l1 ;
List<int>::iterator a , b , c , d , e ;
a = l1 .begin();
b = l0 .begin();
c = l1 .begin();
d = l1 .begin();
e = l1 .begin();
e --;
cout<< l1 <<endl;
a ++;
b = l0 .begin();
e ++;
-- d ;
c ++;
c --;
-- c ;
++ e ;
e --;
l0 .push_front( -3 );
b --;
-- b ;
e --;
cout<< l0 .size()<<endl;
cout<< l0 <<endl;
e --;
++ d ;
b .hook();
l0 .push_front( -1 );
++ b ;
a ++;
c ++;
e --;
-- d ;
c --;
++ a ;
l1 .insert( e , -3 );
-- a ;
-- c ;
-- e ;
c .hook();
cout<< l0 <<endl;
l0 .insert( b , -2 );
```

```
e = l0 .end();
-- c ;
a .hook();
cout<<* a <<endl;
l0 .push_front( -2 );
l1 .push_front( -2 );
-- b ;
++ e ;
l1 .insert( d , 0 );
e ++;
l1 .push_back( 4 );
-- c ;
a --;
l0 .push_front( 3 );
-- c ;
-- d ;
cout<< l1 <<endl;
-- e ;
l1 .insert( d , 1 );
cout<<* e <<endl;
++ d ;
c ++;
a .stopHook();
cout<<* a <<endl;
cout<<* c <<endl;
cout<< l0 .size()<<endl;
cout<< l1 <<endl;
e .hook();
e ++;
cout<< l1 <<endl;
cout<< l1 <<endl;
d --;
e = l1 .begin();
d ++;
a = l1 .end();
cout<< l1 <<endl;
d --;
e ++;
d .hook();
c = l0 .begin();
cout<< l1 <<endl;
e ++;
e --;
```

```
-- b ;
cout<< l0 <<endl;
cout<< l0 .size()<<endl;
cout<< l1 <<endl;
b --;
d = l1 .end();
l0 .push_back( 3 );
l0 .push_front( -2 );
a = l0 .begin();
c .hook();
l1 .push_back( 2 );
++ e ;
e --;
c --;
l1 .push_front( -1 );
a .hook();
e --;
++ b ;
d --;
d ++;
l1 .insert( e , 4 );
l0 .push_front( -2 );
c ++;
l0 .insert( a , -2 );
cout<<* a <<endl;
l0 .push_front( 1 );
cout<< l0 .size()<<endl;
-- a ;
cout<<* b <<endl;
a ++;
e --;
l1 .push_front( -3 );
++ e ;
l0 .insert( b , 0 );
d --;
e ++;
l0 .insert( b , -2 );
c ++;
c ++;
l0 .push_front( -1 );
-- d ;
d ++;
++ b ;
cout<<* b <<endl;
c = l1 .begin();
l1 .insert( d , 0 );
c --;
a ++;
d = l1 .end();
b ++;
a ++;
a ++;
l0 .push_front( -4 );
d --;
++ d ;
cout<< l1 <<endl;
e --;
++ c ;
l1 .insert( e , -2 );
l1 .insert( d , 3 );
l0 .insert( a , 3 );
-- e ;
l0 .push_back( 4 );
cout<< l0 <<endl;
a ++;
e ++;
++ c ;
++ e ;
c --;
l0 .push_front( -3 );
l0 .push_front( -2 );
cout<< l0 <<endl;
b = l0 .begin();
cout<< l0 .size()<<endl;
cout<< l0 .size()<<endl;
d --;
cout<< l1 <<endl;
b .hook();
l1 .insert( c , -3 );
++ b ;
++ a ;
e --;
b .stopHook();
d .hook();
d ++;
++ e ;
cout<< l0 <<endl;
-- c ;
-- d ;
l0 .push_front( 4 );
-- e ;
l1 .insert( c , 0 );
l1 .push_back( -4 );
++ d ;
-- e ;
d ++;
cout<<* c <<endl;
cout<<* e <<endl;
++ c ;
cout<< l1 <<endl;
cout<< l0 <<endl;
l0 .push_front( 3 );
cout<< l1 <<endl;
++ c ;
-- b ;
a ++;
cout<< l1 .size()<<endl;
b ++;
e .hook();
cout<< l0 <<endl;
++ a ;
++ c ;
a ++;
-- d ;
-- e ;
a ++;
-- a ;
b --;
a --;
cout<< l0 <<endl;
cout<< l0 <<endl;
l0 .push_front( -2 );
```

```
cout<< l0 <<endl;
cout<< l1 <<endl;
```

Exemple de sortida 2

```
acde
1
-3b,
-1,-3[b],
-3
-3[a][c],-2,0,4d,
-2
-3
-3
5
-2,-3a[c],0,1,4,d
-2,-3a[c],0,1,4,d
-2,-3a[c],0,1,4,d
-2e,-3[c],0,1,4,ad
-2,-3e,0,1,4[d],a
3c,-2,-3[b],-1,-2,
5
-2,-3e,0,1,4[d],a
-2
10
-3
-3
-3c,-1,4,-2,-3e,0,1,4,0,2,d
-4,-1,1,-2,-2,-2,0,3,3,-2[a],-2,-1,-2,-3[b],3,4,
-2,-3,-4,-1,1,-2,-2,-2,0,3,3,-2,-2[a],-1,-2,-3[b],3,4,
18
18
-3,-1c,4,-2,-2,-3e,0,1,4,0,2,3d,
-3,-2b,-4,-1,1,-2,-2,-2,0,3,3,-2,-1,-2[a],-2,-3,3,4,
-3
-2
-3,0,-3,-1c,4,-2e,-2,-3,0,1,4,0,2,-4,3[d],
4,-3,-2b,-4,-1,1,-2,-2,-2,0,3,3,-2,-1,-2[a],-2,-3,3,4,
-3,0,-3,-1c,4,-2e,-2,-3,0,1,4,0,2,-4,3[d],
15
3,4,-3,-2b,-4,-1,1,-2,-2,-2,0,3,3,-2,-1,-2,-2[a],-3,3,
3,4,-3b,-2,-4,-1,1,-2,-2,-2,0,3,3,-2,-1,-2,-3,-2[a],3,
3,4,-3b,-2,-4,-1,1,-2,-2,-2,0,3,3,-2,-1,-2,-3,-2[a],3,
-2,3,4,-3b,-2,-4,-1,1,-2,-2,-2,0,3,3,-2,-1,-2,-3,-2[a]
-3,0,-3,-1,-2c[e],4,-2,-3,0,1,4,0,2,3[d],-4,
```

Observació

Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, on totes les operacions tenen cost constant (excepte l'escriptura de tota la llista per la sortida, que té cost lineal), i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.

Informació del problema

Autor : PRO2