Mètode de la classe llista per a moure l'ultim element a la segona posició X61197_ca

Implementeu un nou mètode de la classe List que tregui l'element situat en ultima posició, i el posi en segona posició de la llista. En altres paraules, si el contingut de la llista és $[a_1, a_2, a_3, \ldots, a_{n-1}, a_n]$, a on els elements es presenten des del principi de la llista fins al final, llavors recol.locar l'últim element de la llista en segona posició d'aquesta la deixa així: $[a_1, a_n, a_2, a_3, \ldots, a_{n-1}]$. El els casos especials en que la llista té 0 o 1 o 2 elements, el mètode no canvia res.
D'entre els fitxers que s'adjunten en aquest exercici, trobareu `list.old.hpp`, a on hi ha una implementació de la classe genèrica `List`. En primer lloc, haureu de fer:

```
cp list.old.hpp list.hpp
```

A continuació, haureu de buscar dins `list.hpp` la part:

```
// Pre:
// Post: L'últim element de la llista s'ha mogut a segona posició.
//       Si hi havia dos o menys elements a la llista, llavors res ha canviat.
// Descomenteu les següents dues linies i implementeu la funció:
// void moveLastToSecond() {
// }
```

Haureu de descomentar les dues línies que s'indiquen i implementar aquest mètode. No toqueu la resta de la implementació de la classe, excepte si, per algun motiu, considereu que necessiteu afegir algun mètode auxiliar a la part privada.
Preferiblement, haurieu d'aconseguir implementar el mètode a base d'intercanviar els punters de l'objecte. De fet, una implementació que no es basi en això possiblement provocarà error d'execució o una sortida errònia o serà massa lenta.
D'entre els fitxers que s'adjunten a l'exercici també hi ha `program.cpp` (programa principal) i `Makefile` per a compilar. Per a pujar la vostra solució, heu de crear el fitxer `solution.tar` així:

```
tar cf solution.tar list.hpp
```

Entrada

La entrada del programa és una seqüència d'instruccions del següent tipus que s'aniran aplicant sobre una llista que se suposa inicialment buida i un iterador que se suposa situat inicialment al principi (i final) d'aquesta llista:

```
push_front s (s és un string)
push_back s (s és un string)
pop_front
pop_back
it++
it--
*it
print
moveLastToSecond
```

Se suposa que la seqüència d'entrada serà correcta (sense `pop_front` ni `pop_back` sobre llista buida, ni `*it` tenint `it` situat al `end` de la llista). Tampoc hi haurà `pop_front` just quan l'iterador estigui apuntant al primer element de la llista, ni hi haurà `pop_back` just quan l'iterador estigui apuntant a l'últim element de la llista (tingueu en compte que l'últim element de la llista no és el `end` de la llista).
El programa principal que us oferim ja s'encarrega de llegir aquestes entrades i fer les crides als corresponents mètodes de la classe list. Només cal que implementeu el mètode abans esmentat.

Sortida

Per a cada instrucció `*it`, s'escriurà el contingut apuntat per l'iterador. Per a cada instrucció `print` s'escriurà el contingut de tota la llista. El programa que us oferim ja fa això. Només cal que implementeu el mètode abans esmentat.

Exemple d'entrada 1

```
moveLastToSecond
push_front a
it--
*it
moveLastToSecond
*it
push_back b
it++
*it
moveLastToSecond
*it
it--
moveLastToSecond
*it
it++
*it
push_front c
print
moveLastToSecond
*it
push_back d
print
moveLastToSecond
*it
moveLastToSecond
*it
print
it--
*it
it--
*it
it--
*it
print
```

Exemple de sortida 1

```
a
a
b
b
a
b
c a b
b
c b a d
b
b
c a d b
d
a
c
c a d b
```

Exemple d'entrada 2

```
push_back rb
push_back b
it--
pop_front
moveLastToSecond
push_front ow
pop_front
it++
push_front d
it--
```

```
push_back d
moveLastToSecond
push_front o
pop_front
it--
push_front j
pop_front
it--
push_front ef
push_back cb
push_front c
pop_back
push_back x
pop_back
it--
it++
push_back el
pop_back
pop_back
it++
push_back qf
pop_front
it++
moveLastToSecond
push_front m
pop_front
it++
*it
moveLastToSecond
moveLastToSecond
pop_front
push_back e
*it
moveLastToSecond
it++
pop_front
push_back bu
push_back jj
push_front w
it++
it--
*it
push_front xi
moveLastToSecond
*it
pop_back
push_front bl
moveLastToSecond
it--
push_front n
pop_back
pop_front
it--
pop_front
moveLastToSecond
moveLastToSecond
*it
*it
push_back rr
push_back f
```

```
it++
it--
it++
moveLastToSecond
pop_front
it--
push_front fb
*it
it++
it++
it--
*it
print
```

Exemple de sortida 2

```
d
d
e
e
```

```
jj
jj
jj
rr
fb f w e xi jj rr
```

Observació

Avaluació sobre 10 punts: (Afegiu comentaris si el vostre codi no és prou clar)

- Solució lenta: 6 punts.
- solució ràpida: 10 punts.

Entenem com a solució lenta una que és correcta i capaç de superar els jocs de proves públics. Entenem com a solució ràpida una que és correcta i capaç de superar els jocs de proves públics i privats.

Informació del problema

Autor : PRO1
Generació : 2023-05-07 09:13:15