Mètode de llistes per intercanviar (swappejar) el primer i l'últim element X96416_ca

Implementeu un nou mètode de la classe List que intercanvii el primer i l'últim element de la llista. Això vol dir que cadascun d'aquests elements passarà a ocupar la posició dins la llista que ocupava l'altre element. No s'ha de crear ni destruïr memòria. No s'han d'intercanviar els value dels elements. Cal ajustar els seus `next` i `prev` per a que el que era primer passi a ser últim i el que era últim passi a ser primer.
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 de buscar dins `list.hh` les següents línies:

```
// Pre:
// Post: L'element que era el primer de la llista ha passat a ser l'últim de la
//       L'element que era l'últim de la llista ha passat a ser el primer de la
//       A part d'això, res més ha canviat.
//       No s'ha creat ni eliminat memòria.
//       En els casos particulars en que hi havien 0 o 1 elements a la llista,
// Descomenteu les següents dues linies i implementeu el mètode:
// void swapFirstLast() {
// }
```

Descomenteu les dues linies que s'indiquen i implementeu el 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.
Nota: pot ajudar tractar a part el cas particular en que teniu exactament dos elements a la llista.
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

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
swapFirstLast
```

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 els mètodes abans esmentats.

Sortida

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

Exemple d'entrada 1

```
swapFirstLast
push_front a
it--
*it
swapFirstLast
*it
push_back b
it++
*it
swapFirstLast
*it
it++
*it
swapFirstLast
*it
it++
*it
push_front c
*it
it--
*it
it--
*it
swapFirstLast
*it
it--
*it
swapFirstLast
*it
it++
*it
swapFirstLast
*it
it++
*it
it++
*it
```

Exemple de sortida 1

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

Exemple d'entrada 2

```
push_front de
swapFirstLast
it--
push_back q
swapFirstLast
```

```
swapFirstLast
push_front e
push_back p
push_front tf
it--
pop_front
```

```
*it
push_front rq
push_front s
it--
it++
it--
it--
push_back w
push_back ou
push_front uu
it++
pop_back
push_back gs
push_back ok
push_front s
it++
swapFirstLast
it++
*it
push_back j
pop_front
push_back xg
push_front lo
push_front j
pop_back
swapFirstLast
push_back i
push_front ir
swapFirstLast
*it
```

Exemple de sortida 2

```
e
de
de
```

Observació

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

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

Informació del problema

Autoria: PRO2

Generació: 2026-01-27T18:57:35.329Z