## Moure elements d'una llista doblement encadenada al final, usant memòria dinàmica X20190_ca

Donada la classe `Llista` que permet guardar seqüències ordenades d'enters amb una llista doblement encadenada, sense fantasma, no circular i amb punt d'interès, cal implementar el mètode

```
void moure_x_darrera(const int &x);
/* Pre: El p.i. es una llista L i i el seu punt d'interes
   referencia a la dreta del tot */
/* Post: El p.i. s'ha modificat movent totes les aparicions
   d'x al final, mantenint la resta d'elements en el seu
   ordre; el punt d'interes referencia la primera aparicio d'x,
   si hi era, altrament el punt d'interes no canvia */
```

que desplaça tots els elements iguals a $x$ de la llista paràmetre implícit al final, mantenint l'ordre de la resta d'elements de la llista. Si la llista conté algun element igual a $x$, el punt d'interès referencia a la primera aparició de $x$ dins la llista; altrament, el punt d'interès no canvia.

Els nodes de la classe `Llista` estan doblement encadenats amb punters al següent (`seg`) i a l'anterior (`ant`). Una llista té quatre atributs: la `longitud` i tres punters a nodes, un pel primer element (`primer_node`), un per l'últim (`ultim_node`) i un altre per l'element actual (`act`), on tenim situat el punt d'interès de la llista.

### Entrada

Com a entrada hi haurà una llista, amb el punt d'interès situat a la dreta de tot, i a continuació hi haurà un o més enters addicionals. Per l'entrada de la llista es donen el nombre d'enters $n$ de la llista i $n$ els enters que la formen.

Per llegir les llistes, s'ha utilitzat l'operador >> que es troba definit a la classe `Llista`.

### Sortida

Com a sortida es mostrarà la llista original. A continuació, per cada enter $x$ addicional, es desplaçaran els elements iguals a $x$ al final de la llista i es mostrarà la llista resultant. D'un enter $x$ al següent, s'usarà la llista modificada en el pas anterior.

Per escriure una llista, es recorren i mostren els seus elements del primer a l'últim, i després de l'últim al primer. L'element referenciat pel punt d'interès, en cas que n'hi hagi, apareix entre parèntesis. Per escriure les llistes, s'ha utilitzat l'operador << que es troba definit a la classe `Llista`.

### Observació

Feu la solució usant els atributs privats de la classe `Llista` en lloc dels mètodes públics.

Heu d'enviar la solució comprimida en un fitxer .tar:

```
tar cvf program.tar llista_moure_x_darrera.cpp
```

Observeu que per compilar us donem el `Makefile`, la classe `Llista` amb tots els seus mètodes implementats excepte `moure_x_darrera` i el programa principal `program.cpp`.

### Exemple d'entrada

```
7
1 5 1 7 0 7 1
7
0
3
1
```

### Exemple de sortida

```
[1,5,1,7,0,7,1]>
[1,7,0,7,1,5,1]<

[1,5,1,0,1,(7),7]>
[7,(7),1,0,1,5,1]<

[1,5,1,1,7,7,(0)]>
[(0),7,7,1,1,5,1]<

[1,5,1,1,7,7,(0)]>
[(0),7,7,1,1,5,1]<

[5,7,7,0,(1),1,1]>
[1,1,(1),0,7,7,5]<
```

### Informació del problema

Autor : Neus Català
Generació : 2020-06-12 17:06:23