++ i – amb rebot a la classe List — X51866_ca

Típicament, l'operador ++ dels iteradors de la classe List els desplaça una unitat cap al end de la llista, i l'operador −− dels iteradors de la classe List els desplaça una unitat cap al begin de la llista. A més a més, 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.
En aquest exercici modificarem la classe List de manera que els errors d'execució abans esmentats ja no es produiran. En canvi, es produirà un intercanvi en la direcció de moviment dels operadors ++ i −− (efecte rebot). Per exemple, si creem un iterador nou, el col.loquem al end de la llista, i executem ++ sobre ell, no hi haurà error d'execució, l'iterador no es mourà, i a partir d'aquell moment l'operador ++ sobre ell l'anirà desplaçant cap al begin de la llista, i l'operador −− el desplaçarà cap al end de la llista.
Fixeu-vos en el següent exemple de programa i el seu comportament descrit en els seus comentaris.

```
List<string> l;                          // l:
l.push_back("a");                        // l: a
l.push_back("b");                        // l: a,b
l.push_back("c");                        // l: a,b,c
List<string>::iterator it = l.begin(); // l: (a),b,c
it++;                                    // l: a,(b),c
it++;                                    // l: a,b,(c)
it--;                                    // l: a,(b),c
it++;                                    // l: a,b,(c)
it++;                                    // l: a,b,c,()
it++;                                    // l: a,b,c,()
it++;                                    // l: a,b,(c)
it++;                                    // l: a,(b),c
it--;                                    // l: a,b,(c)
it++;                                    // l: a,(b),c
it++;                                    // l: (a),b,c
it++;                                    // l: (a),b,c
it++;                                    // l: a,(b),c
it--;                                    // l: (a),b,c
it--;                                    // l: (a),b,c
it--;                                    // l: a,(b),c
it--;                                    // l: a,b,(c)
it++;                                    // l: a,(b),c
it--;                                    // l: a,b,(c)
it--;                                    // l: a,b,c,()
it--;                                    // l: a,b,c,()
it++;                                    // l: a,b,c,()
it--;                                    // l: a,b,(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`. Caldrà que modifiqueu la classe `List` per a poder

recordar quina és la direcció actual de desplaçament d'un iterador donat, i reimplementeu els operadors ++ i -- convenientment. Més específicament, haureu de buscar les següents línies i fer els afegits i modificacions que s'hi indiquen:

```
  ...

  // Iterators mutables
  class iterator {
    friend class List;
    private:
      List *plist;
      Item *pitem;
      // Add an attribute to remember the orientation of the iterator:
      // ...

      // You can add new private methods if you wish.

    public:

    iterator() {
        // Initialise the orientation of the iterator
        // ...
    }

    // Adapt this function so that no error occurs and the orientation of the i
    // is taken into account and updated accordingly.
    // 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: ++ on iterator at the end of list" << endl;
        exit(1);
      }
      pitem = pitem->next;
      return *this;
    }

...

    // Adapt this function so that no error occurs and the orientation of the i
    // is taken into account and updated accordingly.
    // 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: -- on iterator at the beginning of list" << endl;
       exit(1);
      }
      pitem = pitem->prev;
      return *this;
    }

...
```

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 té una seqüència d'instruccions del següent tipus que s'aniran aplicant sobre la llista i dos iteradors que se suposen situats inicialment al principi (i final) de la llista:

```
push_front s (s és string)
push_back s (s és string)
pop_front
pop_back
it1 = begin
it1 = end
it1 = erase it1
it1++
it1--
++it1
--it1
*it1 = s (s és string)
insert it1 s (s és string)
cout << *it1
it2 = begin
it2 = end
it2 = erase it2
it2++
it2--
++it2
--it2
*it2 = x (x és string)
insert it2 x (x és string)
cout << *it2
cout << l
```

Se suposa que la seqüència d'entrada serà correcta, és a dir, que no es produeixen errors d'execució si s'apliquen correctament sobre una llista i dos iteradors amb les condicions abans esmentades.

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 `sum` abans esmentat.

Sortida

Per a cada instrucció `cout << *it1` o `cout << *it2` s'escriurà el contingut apuntat per l'iterador `it1` o `it2`, respectivament. Per a cada instrucció `cout << l` s'escriurà el contingut de tota la llista. El programa que us oferim ja fa això. Només cal que feu els canvis abans esmentats.

Exemple d'entrada 1

```
cout << l
it1--
--it1
it1++
++it1
it2--
--it2
it2++
cout << l
push_back a
push_back b
push_back c
cout << l
it1 = begin
cout << l
it1--
cout << l
--it1
cout << l
it1--
cout << l
++it1
cout << l
--it1
cout << l
it1--
cout << l
it1++
cout << l
it1--
cout << l
it1--
cout << l
it1--
cout << l
it1++
cout << l
it1++
cout << l
it1++
cout << l
it1--
cout << l
it1 = begin
it2 = end
cout << l
it2++
cout << l
it2++
cout << l
cout << *it1
cout << *it2
push_front i
push_back j
push_front k
push_back l
cout << l
it1 = end
it1++
it1--
--it1
it1--
it2++
++it2
it2--
--it2
--it2
cout << l
it1 = begin
it2 = end
it2++
++it2
cout << l
cout << *it1
cout << *it2
it2++
++it2
it1--
it1--
cout << l
cout << *it1
cout << *it2
insert it1 o
insert it2 p
cout << l
pop_front
pop_back
cout << l
it1--
--it1
it2++
cout << l
it1 = erase it1
cout << l
```

```
++it2
cout << l
it2 = erase it2
cout << l
*it1 = x
cout << l
*it2 = y
cout << l
it1++
++it2
cout << l
it1--
it2--
cout << l
```

Exemple de sortida 1

```
([])
([])
a b c ([])
(a) b c []
(a) b c []
a (b) c []
a b (c) []
a (b) c []
a b (c) []
a b c ([])
a b (c) []
a b c ([])
a b c ([])
a b (c) []
a b c ([])
a b c ([])
a b (c) []
a b c ([])
(a) b c []
(a) b c []
(a) b [c]
a
c
k i (a) b [c] j l
k i a b c ([j]) l
(k) i a b c j [l]
k
l
k (i) a b [c] j l
i
c
k o (i) a b p [c] j l
o (i) a b p [c] j
o i a (b) [p] c j
o i a ([p]) c j
o i [a] (p) c j
o i ([p]) c j
o i ([x]) c j
o i ([y]) c j
o ([i]) y c j
o i ([y]) c j
```

Exemple d'entrada 2

```
push_front bi
it2--
it2 = begin
cout << *it2
++it1
it2++
it2--
cout << *it2
push_back yqs
it2++
it2 = end
insert it2 gfn
--it1
it2 = begin
++it2
```

```
pop_back
push_front lf
push_back t
it2 = end
push_front dec
push_front tj
it1--
++it1
push_back p
push_front am
push_front rpqe
it2++
push_front ssg
--it2
push_back e
push_front y
push_back pgjh
```

```
insert it1 flh
it2--
push_back lmfd
--it2
insert it1 e
push_front p
insert it1 rq
cout << *it2
--it1
cout << l
cout << *it1
push_front tm
cout << *it2
cout << *it1
pop_front
*it2 = wrl
it1++
it1--
insert it2 gfou
++it1
++it1
it1 = end
insert it2 wp
push_front a
push_front g
it2++
it2++
push_front g
it1--
++it2
++it1
push_back dlx
push_front rrg
it2++
++it1
it2++
push_front a
insert it2 jnec
insert it2 ac
++it2
++it2
push_front d
cout << l
it1--
push_front zq
insert it1 xy
--it2
--it2
cout << l
it1--
++it1
it2--
push_front gln
insert it1 k
--it1
cout << *it2
it2++
it1++
++it1
--it2
++it1
```

```
it1++
cout << *it2
pop_back
it2 = end
cout << *it1
--it1
--it1
push_back pkfc
push_front qivz
push_front yhj
it2--
push_front av
--it1
cout << *it1
++it1
cout << *it2
it1++
it2 = begin
++it2
push_back ywb
pop_back
--it1
it1++
insert it2 nma
it2 = erase it2
++it2
push_back vvv
--it2
cout << *it2
++it1
push_front j
insert it1 cn
push_back rca
*it2 = s
push_back ouv
push_front mgm
insert it1 trm
cout << *it1
it2 = begin
++it1
it2--
*it1 = a
insert it1 ya
it1 = begin
cout << *it2
push_front fw
insert it1 xzm
push_back ibxw
++it2
it2 = erase it2
insert it2 mkru
it1++
push_front vb
++it1
--it1
--it2
it2--
--it1
cout << *it1
it2--
insert it1 e
```

```
cout << *it1
it1--
push_front iksl
insert it2 bgsb
cout << *it2
push_front s
cout << *it1
++it1
push_back ios
push_back pkls
*it2 = j
it1++
cout << *it1
insert it1 mzrn
++it2
cout << *it1
push_back lyfe
cout << *it2
cout << *it2
push_front fg
--it1
cout << *it1
it1--
cout << *it1
push_back l
push_front j
push_front ff
it2++
push_front yu
push_back i
cout << *it1
it2++
cout << *it2
cout << *it2
++it1
push_front xc
insert it2 yw
it1--
push_back mrjl
it1++
it1++
it2++
cout << l
it1++
push_front e
push_front ynuk
--it2
*it1 = ir
++it2
*it2 = zv
cout << *it2
it2--
--it2
++it2
push_back d
insert it2 ot
push_back tas
insert it1 mk
push_back yn
it2 = end
push_back vwp
```

```
push_front vdwx
it2--
it1++
insert it2 czie
cout << *it1
cout << *it2
insert it2 zqf
++it2
++it2
cout << *it1
push_back tl
cout << *it1
cout << *it1
push_front en
insert it1 li
push_back ual
cout << *it1
*it1 = v
cout << l
cout << l
push_front wz
it2--
++it1
it2++
++it2
push_front p
push_back ki
--it2
++it2
*it2 = hvy
cout << l
it2 = erase it2
it1++
cout << *it1
it1--
cout << *it1
it2--
++it1
*it1 = lnel
insert it1 mm
--it1
it2--
--it2
--it2
push_front cf
push_back ilaw
++it2
push_back esz
push_front wv
--it2
cout << *it2
--it1
insert it2 zi
++it1
--it1
cout << *it1
cout << *it1
insert it1 i
it2++
cout << l
cout << l
```

```
cout << *it1
pop_back
--it2
push_back sq
push_back r
cout << l
insert it2 i
insert it1 bzt
it2++
*it1 = zoay
push_back cc
it2 = erase it2
++it2
it2++
cout << *it1
it1++
it1--
cout << *it1
it2--
push_back rj
++it1
cout << *it1
push_back wku
push_back ura
--it2
push_front secy
insert it2 t
it2++
--it2
it2++
--it1
insert it2 ovn
it1 = begin
--it2
cout << *it1
--it2
push_front yfpn
it1++
insert it2 uc
insert it1 b
pop_front
--it2
insert it2 yx
--it1
it1 = begin
cout << *it2
push_back sasc
push_back q
cout << *it2
it2++
++it1
*it2 = g
push_front hdvm
pop_back
it2 = begin
it2--
*it2 = mr
*it2 = i
it2++
--it2
it1++
```

```
insert it1 tky
push_back j
it2 = erase it2
push_front r
it2--
push_front uio
push_front pnxv
++it1
cout << *it2
cout << l
*it1 = h
insert it2 lte
push_front mo
insert it2 g
it1--
--it2
insert it1 jrh
it1--
cout << *it1
cout << *it2
it2--
--it1
push_front keb
it1--
cout << *it1
push_front zp
cout << *it2
cout << l
```

Exemple de sortida 2

```
bi
bi
pgjh
p y ssg rpqe am tj dec lf bi yqs t p e [pgjh] flh lmfd e (rq)
rq
pgjh
rq
d a rrg g g a p y ssg rpqe am tj dec lf bi yqs t p e gfou wp wrl flh lmfd e rq jnec ac dlx ([])
zq d a rrg g g a p y ssg rpqe am tj dec lf bi yqs t p e gfou wp wrl flh lmfd e rq jnec ac dlx [xy
dlx
dlx
xy
pkfc
pkfc
qivz
vvv
mgm
mgm
mgm
nma
e
j
j
bgsb
bgsb
mzrn
mgm
mgm
j
j
xc yu ff j fg s iksl vb fw mkru e mgm mzrn ([yw]) j av bgsb j s gln zq d a rrg g g a p y ssg rpqe
```

```
zv
av
vwp
av
av
av
av
en vdwx ynuk e xc yu ff j fg s iksl vb fw mkru e mgm mzr
en vdwx ynuk e xc yu ff j fg s iksl vb fw mkru e mgm mzr
p wz en vdwx ynuk e xc yu ff j fg s iksl vb fw mkru e mg
j
bgsb
tl
bgsb
bgsb
wv cf p wz en vdwx ynuk e xc yu ff j fg s iksl vb fw mkr
wv cf p wz en vdwx ynuk e xc yu ff j fg s iksl vb fw mkr
bgsb
wv cf p wz en vdwx ynuk e xc yu ff j fg s iksl vb fw mkr
zoay
zoay
mm
secy
uc
uc
b
pnxv uio r secy [b] tky wv (cf) p wz en vdwx ynuk e xc
jrh
tky
b
jrh
zp keb mo pnxv uio r secy lte g (b) tky [jrh] wv h p wz
```

Observació

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

- Solució lenta: 5 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

Autoria: PRO2

Generació: 2026-01-27T18:53:06.689Z