```
template <class T> class list {

// Tipus de mòdul : dades
// Descripció del tipus: Estructura lineal que conté elements de tipus T,
// que es pot començar a consultar pels extrems, on des de cada element es
// pot accedir a l'element anterior i posterior (si existeixen), i que
// admet afegir-hi i esborrar-hi elements a qualsevol punt

// El cost temporal de totes les operacions és constant, tret de la copiadora,
// la destructora i clear, que tenen cost lineal respecte a la mida de la llista
// original (en aquestes, i en insert i erase, també cal tenir en compte
// el cost de la còpia o esborrat de cada objecte implicat de tipus T)

private:

public:

// Constructores

list();
/* Pre: cert */
/* Post: El resultat és una llista sense cap element */

list(const list& original);
/* Pre: cert */
/* Post: El resultat és una llista còpia d'original */


// Destructora: Esborra automàticament els objectes locals en sortir d'un
// àmbit de visibilitat

~list();


// Modificadores

void clear();
/* Pre: cert */
/* Post: El paràmetre implícit és una llista buida */

void insert(iterator it, const T& x);
/* Pre: it referencia algun element existent al paràmetre implícit o
   és igual a l'end d'aquest */
/* Post: El paràmetre implícit és com el paràmetre implícit original amb x
   davant de l'element referenciat per it al paràmetre implícit original */

iterator erase(iterator it);
/* Pre: it referencia algun element existent al paràmetre implícit,
   que no és buit */
/* Post: El paràmetre implícit és com el paràmetre implícit original sense
   l'element referenciat per l'it original; el resultat referencia
   l'element següent al que referenciava it al p.i. original */

// Nota: si volem que el propi it passi a referenciar l'element següent
   al que referenciava al p.i. original d'una llista l, hem de cridar
   it = l.erase(it)

void splice(iterator it, list& l);
/* Pre: l=L, it referencia algun element del paràmetre implícit o
   és igual a l'end d'aquest, l i el p.i. no són el mateix objecte */
/* Post:  S'han transferit al paràmetre implícit els elements d'L, inserits
   abans de l'element referenciat per it; l és buida */
```

```
// Consultores

bool empty() const;
/* Pre: cert */
/* Post: El resultat indica si el paràmetre implícit té elements o no */

int size() const;
/* Pre: cert */
/* Post: El resultat és el nombre d'elements del paràmetre implícit */


// Iteradors típics

iterator begin();
/* Pre: cert */
/* Post: El resultat és un iterator al principi del paràmetre implícit */

const_iterator begin() const;
/* Pre: cert */
/* Post: El resultat és un const_iterator al principi del paràmetre implícit */

iterator end();
/* Pre: cert */
/* Post: El resultat és un iterator a un element fictici immediatament posterior
al final del paràmetre implícit */

const_iterator end() const;
/* Pre: cert */
/* Post: El resultat és un const_iterator a un element fictici immediatament
posterior al final del paràmetre implícit */

// Notes:
// a) si l és buida, l.begin() és el mateix que l.end()
// b) si l ve qualificada com a const, l.begin() i l.end() retornen un
// const_iterator; en cas contrari, retornen un iterator
};


/* Operacions amb iterators:

++it : Avança al següent element, no vàlid a l'end

--it : Retrocedeix a l'anterior element, no vàlid al begin

*it : Designa l'element referenciat per it; no vàlid per a l'end o per a
      iteradors que no referencien res; si la llista d'it ve qualificada
      com a const o si it és un const_iterator, llavors *it és "read-only"

it1=it2 : Assigna l'iterador it2 a it1; un const_iterator no es pot assignar a
un iterator

it1==it2 : val true si els iteradors it1 i it2 són iguals; false si no

it1!=it2 : val true si els iteradors it1 i it2 són diferents; false si no

*/
```