```
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

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 l'original amb x
   davant de l'element referenciat per it en el 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 l'original sense
   l'element referenciat per l'it original; el resultat referencia l'element
   següent al referenciat per it en el paràmetre implícit original */

void splice(iterator it, list& l);
/* Pre: l=L, it referencia algun element del paràmetre implícit o
   és igual a l'end() d'aquest */
/* Post:  El paràmetre implícit té els seus elements originals i els
   de 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 constants

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 */

// Nota: si l és buida, l.begin() és el mateix que l.end()

};


/* 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 a l'end o a iteradors que
no referencien res

it1=it2 : Assigna l'iterador it2 a it1

it1==it2 : Diu si els iteradors it1 i it2 són iguals o no

it1!=it2 : Diu si els iteradors it1 i it2 són diferents o no

*/
```