FONDAMENTI DI PROGRAMMAZIONE **PROVA PRATICA 07 GEN 2025**

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

L'obiettivo di questa prova pratica è implementare una lista di caratteri tramite una classe chiamata `ListaCaratteri`. La classe deve gestire una lista di caratteri di numero potenzialmente illimitato.

Implementare le seguenti operazioni che possono essere effettuate su `ListaCaratteri`:

--- Metodi invocati nella PRIMA PARTE di main.cpp: ---

✓ `ListaCaratteri lc;`

Costruttore di default che inizializza una `ListaCaratteri`, inizialmente vuota.

✓ `lc.inserisci(carattere, fine);`

Inserisce un nuovo carattere nella lista. Se `fine` è true il carattere viene inserito in coda, altrimenti in testa alla lista.

✓ `lc.rimuovi(c, tutti);`

Rimuove il primo carattere uguale a c dalla lista. Se `tutti` è true rimuove anche tutte le successive occorrenze di quel carattere nella lista. Ritorna true se almeno un carattere è stato rimosso.

✓ `cout << lc;`

Operatore di uscita per il tipo `ListaCaratteri`, che stampa a schermo la lista nel seguente formato:

```
a -> b -> c -> @
```

L'esempio mostra la lista di caratteri composta dai caratteri 'a', 'b' e 'c', inseriti in questo esatto ordine. Il carattere "@" viene SEMPRE stampato per indicare la fine della lista, a prescindere dalle sue dimensioni. NOTA 1: c'è uno spazio tra ogni carattere e la coppia di caratteri "->". NOTA 2: La lista vuota viene stampata con il solo simbolo "@".

--- Metodi invocati nella SECONDA PARTE di main.cpp: ---

✓ `~ListaCaratteri;`

Implementare il distruttore, qualora necessario.

✓ `lc1 == lc2;`

Controlla se due liste sono uguali. Due liste sono uguali se hanno lo stesso numero di caratteri e i caratteri appaiono nello stesso ordine in entrambe le liste.

✓ `~lc;`

Operatore di negazione bitwise, inverte l'ordine degli elementi nella lista. Esempio, data la lista caratteri `lc`

```
a -> b -> c -> @
```

Dopo l'applicazione dell'operatore di negazione bitwise la lista caratteri `lc` sará:

```
c -> b -> a -> @
```

✓ `lc.controllaPalindroma();`

Verifica se la lista è palindroma. Restituisce `true` se lo è, `false` altrimenti.

✓ `lc.cercaSottostringa(lc2);`

Verifica se la lista caratteri passata come parametro è una sottostringa della lista principale, ritorna `true` in caso positivo, `false` altrimenti. Una lista di caratteri è una sottostringa di un'altra se i caratteri della prima compaiono consecutivamente, nello stesso ordine all'interno della seconda. Esempio: la stringa "adu" è sottostringa di "caduto".

✓ `lc.estraiNultimoCarattere(n);`

Estrae l'ennesimo carattere dalla lista partendo dal fondo. Il carattere viene eliminato dalla lista e un puntatore al carattere viene restituito al chiamante. Esempio, data la lista di caratteri **`lc:`**

`a -> b -> c -> d -> e -> f -> g -> @`

Se chiamiamo il metodo **`lc.estraiNultimoCarattere(3)`**, ci aspettiamo di dover estrarre il carattere **`'e'`**, in quanto è il terzo carattere a partire dal fondo della lista. La lista di caratteri risultante è dunque:

`a -> b -> c -> d -> f -> g -> @`

Pertanto l'elemento che contiene il carattere **`'e'`** deve essere rimosso dalla lista, mentre la funzione deve ritornare al chiamante un puntatore ad una variabile carattere contenente una copia del carattere estratto (il carattere **`'e'`** in questo caso).

Mediante il linguaggio C++, realizzare il tipo di dato astratto **`ListaCaratteri`**, definito dalle precedenti specifiche. Non è permesso utilizzare funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc. **Gestire le eventuali situazioni di errore**.

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---
Lista iniziale vuota: @
Dopo l'inserimento in coda (a, b, c): a -> b -> c -> @
Dopo l'inserimento in testa (z): z -> a -> b -> c -> @
Rimozione caratteri: 110
Dopo la rimozione: a -> c -> @
Lista prima della rimozione dei caratteri 'a' e 'c': r -> a -> c -> e -> c -> a -> r -> @
Rimuovi tutti i caratteri 'a' e 'c': r -> e -> r -> @

--- SECONDA PARTE ---
Controllo inverso lista: Si
Controllo lista palindroma: Si
Lista da cercare: c -> a -> d -> b -> @
Sottostringa da trovare: a -> d -> @
La sottostringa fa parte della lista? Si
Estrazione del terzo elemento dalla fine: e
Lista dopo l'estrazione: a -> b -> c -> d -> f -> g -> @
```

Note per la consegna:

Affinché l'elaborato venga considerato valido, il programma **deve** produrre almeno la prima parte dell'output atteso. In questo caso, i docenti procederanno alla valutazione dell'elaborato **solo se** lo studente avrà completato l'autocorrezione del proprio elaborato.
In **tutti** gli altri casi (per esempio, il programma non compila, non collega, non esegue o la prima parte dell'output non coincide con quella attesa), l'elaborato è considerato **insufficiente** e, pertanto, **non verrà corretto**.