Compito di Programmazione del 20/9/2016

Abbiamo usato molte volte la struttura FIFO che consiste di 2 campi primo e ultimo che puntano al primo e all'ultimo nodo di una lista concatenata in modo da rendere facile eseguire operazioni sia all'inizio che alla fine della lista. Ora vogliamo estendere FIFO in una struttura FIFOX che oltre ai campi primo e ultimo possiede 3 campi interi**: valp, valu e nele**. Questi 3 campi contengono, rispettivamente, il valore del campo info del primo nodo (quello puntato da primo), il valore valore del campo info dell'ultimo nodo (quello puntato da ultimo) e il numero di nodi presenti nella lista. L'idea è che un valore FIFOX venga usato per gestire una lista concatenata ordinata secondo i campi info dei nodi. In questo modo, usando i campi valp e valu diventa facile sapere se un nodo con un certo valore può essere nella lista anche senza percorrere la lista stessa. Il valore FIFOX per gestire una lista vuota ha tutti i 5 campi a 0.

Esempio 1. Supponiamo di avere la seguente lista ordinata: -4->-2->0->10, allora il valore FIFOX A per gestire questa lista sarebbe: valp=-4, valu=10, nele=4 e primo e ultimo puntano, rispettivamente al primo (-4) e all'ultimo (10) nodo della lista. Diremo che un tale valore FIFOX A è **corretto** (rispetto alla lista che gestisce).

Ovviamente quando vogliamo aggiungere un nuovo nodo ad una lista gestita con un valore FIFOX A, dobbiamo avere cura di generare un nuovo valore FIFOX che corrisponda alla nuova lista.

Esempio 2. Supponiamo di voler aggiungere un nuovo nodo con info=11 alla lista dell'esempio 1, allora, per mantenere l'ordine, la nuova lista deve essere: -4->-2->0->10->11 e il nuovo valore FIFOX per questa nuova lista sarebbe: primo che punta a -4, ultimo a 11, valp=-4, valu=11, nele=5.

Esercizio 1 (10 punti): scrivere una funzione iterativa FIFOX push_iter(FIFOX A, nodo*b) che soddisfi le seguenti PRE e POST:

PRE=(A è un valore FIFOX corretto che gestisce una lista corretta e b è un nodo. Si assume che i nodi della lista abbiano campo info distinti e anche diversi dal campo info di b)

POST=(restituisce un nuovo valore FIFOX corretto che corrisponde alla lista ordinata ottenuta dalla lista di A inserendo il nodo b in essa)

Si considera ora l'operazione di eliminare un nodo da una lista gestita con un FIFOX.

Esempio 3. Consideriamo la lista dell'esempio 2 : -4->-2->0->10->11 e immaginiamo di voler eliminare da questa lista il nodo con -2, la nuova lista corrisponde al valore FIFOX con primo che punta a -4 e ultimo a 11, valp=-4 e valu=11 e nele=4. Insomma il valore FIFOX corretto rispetto alla nuova lista è diverso da quello della lista originale solo per il campo nele. Se invece volessimo eliminare il nodo -4, allora il nuovo valore FIFOX sarebbe: primo che punta a -2, ultimo a 11, valp=-2, valu=11 e nele=4. Infine se volessimo eliminare un nodo che non c'è, per esempio un nodo con valore 5, allora la lista non cambierebbe e di conseguenza, non cambierebbe neppure il corrispondente valore FIFOX. Per sapere che questo è il caso dovremmo percorrere la lista, ma se volessimo eliminare un nodo con info=-10 oppure con info=20, allora, usando i campi valp e valu, sapremmo che queste operazioni non cambiano la lista senza percorrerla.

Esercizio 2(8 punti) Si chiede di scrivere una funzione : FIFOX deleteX(FIFOX A, int z) che soddisfa le seguenti pre e post-condizioni:

PRE_deleteX=(A è un FIFOX corretto e z è un intero. La lista gestita da A ha i nodi con info distinti)

POST_deleteX=(restituisce un FIFOX corretto rispetto alla lista ottenuta da quella di A dopo aver eliminato un nodo con info=z, se un tale nodo esiste e altrimenti restituisce A)

Attenzione: deleteX e le eventuali funzioni ausiliarie che essa invoca, possono essere ricorsive o no, ma in ogni caso **non possono** usare cicli iterativi.

Correttezza:

1) **(3 punti)** dare un invariante del ciclo principale di push_iter

2**) (5 punti)** dimostrare per induzione la correttezza di del_ric.

Il programma dato: La dichiarazione di FIFOX, del suo costruttore e anche dell'operazione di stampa di un valore FIFOX sono contenuti nel programma dato. Notate che l'operazione di stampa è definita sfruttando l'overloading di operator<< per il nuovo tipo FIFOX. Osservate che il main dato esegue 2 cicli di lettura dove il primo serve per testare push_iter e il secondo per testare deleteX. Alla fine di ogni ciclo, il main esegue la stampa del valore FIFOX ottenuto.