Esame di Programmazione del 19/9/2018 tempo 2 ore

Attenzione: se la vostra consegna NON passa i 4 test, allora SICURAMENTE ha qualcosa di sbagliato, se invece li passa, non dovete ASSOLUTAMENTE considerarla una garanzia della sua correttezza. Quando qualcuno dei test fallisce, il moodle mostra l'input solo del primo test che fallisce.

Data una lista concatenata L, un pezzo crescente di L è una sequenza di nodi consecutivi di L tale che i campi info di questi nodi sono uguali o crescenti percorrendoli da sinistra a destra.

Esempio 1: sia L= 2->3->4->-2->-2->3->1->1->10->11->2->0. Un pezzo crescente di L è, 3->4, oppure 1->1->10->11 o anche 2 da solo e in realtà ogni nodo singolo di L è un pezzo crescente di L. In questo esercizio ci interessano i pezzi che non sono allungabili. Per esempio 2->3 è un pezzo, ma è allungabile in 2->3->4. Se consideriamo i pezzi Crescenti e Non Allungabili (CeNA) di L, essi sono: 2->3->4, 2->2->3, 1->1->10->11, 2, e 0.

L'esercizio richiede di smembrare una lista L nei suoi pezzi CeNA e di costruire una lista concatenata LL i cui nodi puntano ai pezzi CeNA di L. I nodi di LL devono avere un tipo struttura nodoL che è definito nel programma dato.

Esempio 2: facendo riferimento all'Esempio 1, il primo nodo di LL punta al pezzo 2->3->4, il secondo nodo al pezzo 2->2->3, il terzo a 1->1->10->11, il quarto e il quinto a 2 e 0, rispettivamente.

Esercizio ricorsivo:

Si tratta di costruire una funzione **ricorsiva** Gric che, data una lista L, costruisca una lista LL i cui nodi puntino ai pezzi CeNA di L come spiegato negli esempi precedenti. Prototipo, PRE e POST di Gric sono come segue:

PRE=(lista(L) ben formata)

nodoL* Gric(nodo*L)

POST=(restituisce una lista concatenata LL di nodi di tipo nodoL con tanti nodi quanti sono i pezzi CeNA di L e tale che il primo nodo di LL ha campo p che punta al primo pezzo CeNA di L, il secondo nodo punta al secondo pezzo CeNA e così via. I pezzi CeNA puntati da LL sono trasformati in liste ben formate mettendo a 0 il campo next del loro ultimo nodo)

ATTENZIONE: Gric non deve fare copie dei nodi di L. Ovviamente Gric dovrà creare i nodi di LL che avranno il tipo nodoL definito nel programma dato. I nodi di L avranno invece il solito tipo nodo, anch'esso dato nel programma.

CONSIGLIO: conviene che Gric usi (almeno) una funzione ausiliaria anch'essa ricorsiva. In questo caso è richiesto definire la PRE e la POST della funzione ausiliaria.

Esercizio iterativo:

Si tratta di scrivere una funzione iterativa, ordina, che prende in input la lista LL prodotta da Gric e la ordina rispetto alla lunghezza dei pezzi puntati dai suoi nodi.

Esempio 3: facendo riferimento alla lista LL dell'Esempio 2, ordina dovrebbe produrre una lista LL' i cui nodi puntano ai pezzi CeNA, 1->1->10->11, 2->3->4, 2->2->3, 2, e 0, in cui i nodi sono ordinati per lunghezza decrescente dei pezzi puntati. Si osservi anche che i pezzi di uguale lunghezza devono conservare la stessa posizione relativa che hanno in LL. Per esempio, visto che il pezzo 2->3->4 in LL precede il pezzo 2->2->3, così deve essere anche nella lista ordinata LL' prodotta da ordina.

La funzione **iterativa** ordina deve obbedire alle seguenti specifiche:

PRE=(lista(LL) ben formata di nodi nodoL)

nodoL* ordina(nodoL* LL)

POST=(restituisce la lista LL' che contiene i nodi di LL ordinati per lunghezza decrescente dei pezzi puntati. I nodi che puntano a pezzi di uguale lunghezza mantengono la loro posizione relativa in LL)

ATTENZIONE: la funzione ordina deve produrre la lista LL' ordinata usando i nodi di LL. Non deve né creare né distruggere nodi.

CONSIGLIO: è utile che la funzione ordina usi (almeno) una funzione ausiliaria anch'essa iterativa. In questo caso è richiesto definire la PRE e la POST della funzione ausiliaria.

Correttezza:

1) Scrivere la prova induttiva di Gric
2) Scrivere l'invariante del ciclo di ordina.