Esame di Programmazione del 19/12/2019 tempo 2 ore e 30 minuti

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.

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 crescenti di lunghezza massima. Per esempio 2->3 è un pezzo crescente, ma non è di lunghezza massima in quanto è possibile allungarlo in 2->3->4. Se consideriamo i pezzi Crescenti di Lunghezza Massima (CLM) 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 CLM e di costruire una lista concatenata X i cui nodi (da sinistra a destra) puntano ai pezzi CLM di L nell'ordine che essi hanno in L. I nodi di X devono avere un tipo struttura nodoP che è definito nel programma dato sul moodle.

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

Esercizio iterativo

Si tratta di costruire una funzione **iterativa** "affetta" che, data una lista L (costituita da nodi di tipo nodo), costruisca una lista X di nodoP che puntino ai pezzi CLM di L come spiegato nell'Esempio 2. Prototipo, PRE e POST di "affetta" sono come segue:

PRE=(lista(L) ben formata)

nodoP* affetta(nodo*L)

POST=(affetta restituisce una lista concatenata X di nodi di tipo nodoP che puntano ai CLM di L, come spiegato nell'Esempio 2 . I CLM puntati dai nodi di X sono trasformati in liste ben formate mettendo a 0 il campo next del loro ultimo nodo)

ATTENZIONE: "affetta" non deve fare copie dei nodi di L. La funzione "affetta" dovrà invece creare i nodi di X che avranno il tipo nodoP.

CONSIGLIO: conviene usare una funzione ausiliaria che tagli la prossima CLM da una lista data non vuota. La funzione ausiliaria deve essere accompagnata da PRE e POST significative.

Esercizio ricorsivo:

Si tratta di scrivere una funzione ricorsiva "ordina" che si occupi di ordinare la lista X prodotta da "affetta" in modo che le liste puntate dai nodi abbiano lunghezza crescente (o uguale) procedendo da sinistra a destra.

Esempio 3: facendo riferimento all'Esempio 2, "ordina" deve produrre da X una nuova lista Y i cui nodi puntano agli stessi CLM di X, ma ordinati in modo crescente. Quindi il primo nodoP di Y punta al CLM 2, il secondo al CLM 0, il terzo al CLM 2->3->4, e poi seguono nodi che puntano ai CLM 2->2->3, e 1->1->10->11. La funzione **ricorsiva** "ordina" deve obbedire alle seguenti specifiche:

PRE=(lista(X) ben formata di nodi nodoP)

nodoP* ordina(nodoP* X)

POST=(restituisce una lista Y che si ottiene da X come spiegato nell'esempio 3)

ATTENZIONE: la lista Y dovrebbe usare gli stessi nodi di tipo nodoP e gli stessi CLM di X.

CONSIGLIO: è utile che la funzione "ordina" usi (almeno) una funzione ausiliaria, anch'essa ricorsiva, che si occupi di inserire un nodoP in una lista ordinata di nodi nodoP. La funzione ausiliaria deve avere una PRE ed una POST.

Correttezza:

1) Scrivere l'invariante del ciclo principale di "affetta".
2) Scrivere la prova induttiva di "ordina"