Esame del 26/6/2017 Turno 2

Dato un albero binario, se consideriamo il percorso infisso sull'albero, esso definisce un ordine totale dei suoi nodi. Si chiede di scrivere funzioni che, dato un albero ed un intero k >0, siano capaci di restituire il nodo numero k dell'albero (secondo l'ordine infisso) e anche il cammino per arrivare al nodo, se il nodo k-esimo c'è. Vediamo un esempio.

Esempio 1. Sia dato l'albero R=a(b(c(_,_),_),d(e(_,f(_,_)),_)), rispetto al percorso infisso (sottoalbero sinistro, radice e poi sottoalbero destro), il nodo 'c' è il nodo 1, 'b' è il 2, 'a' il 3 , 'e' il 4 , 'f' il 5 e 'd' è il 6. Quindi se si chiede il nodo 3 rispetto all'ordine infisso, la risposta è il nodo 'a' e il cammino per arrivarci è [-1], che indica che il nodo cercato è la radice stessa.

Viene dato un main con varie funzioni di stampa. Viene anche data la definizione della struttura nodo che ha un campo in più rispetto al solito. Si tratta del campo intero n il cui scopo verrà spiegato nel punto (2). Si richiede di scrivere 3 funzioni che sono descritte nel seguito:

1) (punti 10) La funzione ricorsiva nodo* cerca1_infix(nodo*r, int& k, int *C) che soddisfi le seguenti PRE e POST:
   PRE=(albero(r) corretto, k>=1, C è un array di interi con un numero di elementi maggiore della profondità di albero(r), vk = k)
   POST=(se albero(r) ha almeno k nodi, allora la funzione restituisce col return il nodo k rispetto all'ordine infisso dei suoi nodi, e inoltre in C restituisce il cammino che da r porta a questo nodo) && (se invece albero(r) ha, diciamo, m nodi, con m <k, allora il valore finale di k sarà vk-m e la funzione restituisce 0 col return)

2) (punti 6) Si chiede di scrivere una funzione ricorsiva void completa(nodo*r) che soddisfi le seguenti PRE e POST:
   PRE=(albero(r) è corretto, valbero(r)=albero(r))
   POST=(albero(r) ha la stessa struttura di valbero(r), ma ogni nodo contiene nel campo n il numero di nodi del sottoalbero radicato in quel nodo, compreso il nodo stesso)

Esempio 2. Consideriamo di nuovo l'albero R=a(b(c(_,_),_),d(e(_,f(_,_)),_)). Allora il campo n del nodo ''a deve diventare 6, quello di 'b' sarà 2, quello 'c' sarà 1, quello di 'd' è 3, di 'e' è 2 e di 'f' è 1.

Attenzione: viene preferita una funzione "completa" che non usi funzioni ausiliarie.

3) (punti 8) Si chiede di scrivere una funzione **iterativa** nodo* cerca2_infix(nodo* r, int k, int*C) che soddisfi le seguenti PRE e POST:
   PRE=(albero(r) è corretto e ogni suo nodo x contiene nel suo campo n il n. di nodi del sottoalbero radicato in x, 1<=k <= r ->n, C è un array di interi con un numero di elementi maggiore della profondità di albero(r))
   POST= (restituisce il nodo k di albero(r) e in C il cammino da r fino a quel nodo)

   Attenzione: la funzione cerca2_infix usa il valore del campo n dei nodi di albero(r) per seguire senza errori il cammino fino al nodo k. Come si può vedere dal main, il campo n della radice è usato per sapere se il nodo cercato è in albero(r) o no, senza nemmeno visitare l'albero. Questo spiega la condizione k<= r ->n nella PRE e il fatto che la POST indichi che il nodo k verrà sempre trovato.

Correttezza: (ogni domanda vale 2 punti)

1) Dimostrare induttivamente che la funzione cerca1_infix è corretta rispetto a PRE e POST.
2) Dimostrare induttivamente che la funzione completa è corretta rispetto a PRE e POST.

3) Dare un invariante (anche intuitivo) del ciclo principale della funzione cerca2_infix.