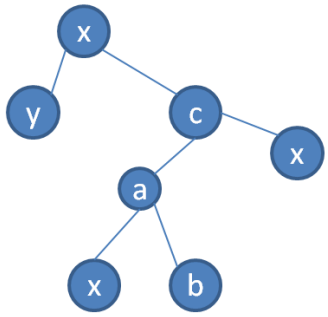
Corso di programmazione 2009-2010

Ottava esercitazione per casa: assegnata il 1 marzo 2010,

Consegna: il 9 Marzo entro le 8 di mattina, con il comando: consegna settimana8

Gli esercizi sono sul pattern matching su alberi binari. Quindi abbiamo un albero binario R (i cui nodi hanno campo informativo char) ed un pattern char* P di dim_P caratteri. Un esempio è utile per capire i concetti in gioco.

Esempio. Sia P=[c,x] ed R punti alla radice del seguente albero binario.



R ha un match contiguo di P che consiste del figlio destro della radice (denotato D) e del suo figlio destro (denotato DD). R ha anche un match non contiguo: sempre il figlio destro della radice e la foglia x che è figlia sinistra del nodo a (questa foglia x è denotata DSS). E' importante capire che consideriamo solo match che giacciono completamente su un cammino dell'albero. Chiameremo **testimone** di un match di P su R una lista concatenata di nodi di tipo struct nodoP {nodo* info; nodoP* next;}; ognuno dei quali punta ad un nodo dell'albero coinvolto da match. Quindi una lista di nodoP il cui primo nodo punti a D ed il cui secondo nodo punti DD è un testimone del match contiguo. Mentre una lista che punti a D e a DSS è un testimone del match non contiguo.

1) Si chiede una funzione ricorsiva con prototipo nodoP* F(nodo* R, char *P, int dim_P) che cerchi un (qualsiasi) match (contiguo oppure non contiguo) di P su un cammino dell'albero R. La funzione deve essere corretta rispetto alle seguenti PRE e POST:

PRE=(R è un valore nodo* valido, dim_P>0 e P[0..dim_P-1] è definito)

POST=(F restituisce K !=0 => esiste un match (contiguo o no) di P in R e la lista K è un testimone di un tale match) &&(F restituisce K=0 => non esiste alcun match (né contiguo né non contiguo) di P in R) &&(F non costruisce alcun nodo nodoP oltre quelli in K)

2) Si chiede una funzione ricorsiva che calcoli il numero totale di match contigui di P su cammini di R. I match possono essere tra loro parzialmente sovrapposti. La funzione deve avere prototipo: int G(nodo*R, char*P, int dim_P). E' consentito introdurre altre funzioni ausiliarie purché ricorsive. La PRE è uguale a quella dell'esercizio (1), la POST scrivetela voi.

Dimostrare la correttezza di entrambe le soluzioni usando l'induzione.