Esame di Programmazione 19/20, quinto appello 16/12/2020. Parte di programmazione.

L'esercizio riguarda il pattern matching, dove il pattern è un array P di dimP interi (dimP>0) e il testo su cui matchare P va costruito, partendo da un albero binario, nel modo seguente.

Dato un albero binario T con nodi del consueto tipo, struct nodo{int info; nodo*left,*right;}; si vuole costruire una lista concatenata C di nodi nodoE come segue: struct nodoE{nodo*info; nodoE* next;};. La lista C deve avere tanti nodi quanti sono i nodi di T e i nodi di C puntano a quelli di T in accordo con l'ordine postfisso dei nodi di T, cioè prima il sottoalbero sinistro, poi quello destro e infine la radice. Il seguente esempio dovrebbe chiarire i dubbi.

Esempio. Sia T= 2(10(_,5(8(_,_),_)), 12(_, 1(6(_,_), 3(_,4(_,_)))), l'ordine postfisso dei nodi di T è il seguente: 8 5 10 6 4 3 1 12 2, quindi C dovrebbe contenere 9 nodi nodoE, dove il primo punta al nodo 8 di T (la foglia più a sinistra), il secondo punta al nodo 5, il terzo al nodo 10 e così via.

Si chiede di scrivere una funzione **ricorsiva** "buildT" che, dato un albero T, costruisca la corrispondente lista C. buildT deve avere la seguente segnatura: nodoE* buildT(nodo*T, nodoE* C) e deve obbedire alle seguenti PRE e POST: PRE=(albero(T) e lista(C) sono ben formate e lista(C)=vC), POST=(restituisce una lista di nodoE che consiste di vC concatenata con la lista di nodoE che punta ai nodi di albero(T) in ordine postfisso).

In secondo luogo, si chiede di scrivere una funzione **iterativa** "match" che cerca il match di lunghezza massima di P sui nodi di T considerati in ordine postfisso (ovviamente usando la lista C costruita nel primo passo). Vanno considerati i match che iniziano in P[0] e poi proseguono con P[1], P[2], ecc, insomma i match contigui, ma non necessariamente completi. Infatti il match può terminare prima della fine di P (a causa di un mismatch). Insomma non è detto che tutto P sia matchato, per questo si cerca il match di lunghezza massima e si deve restituire il puntatore al nodo di C a partire dal quale il match inizia e anche la lunghezza del match trovato. In caso ci fossero vari match con la stessa lunghezza massima, si deve considerare il primo che viene trovato scorrendo C da sinistra. Vediamo un esempio.

La funzione match deve avere il seguente prototipo:

void match (nodoE*C, int*P, int dimP, nodoE*&inizio, int&lung)

Esempio. Supponiamo che la lista C punti a nodi di T con i seguenti valori info: 2 2 5 2 5 2 6 e che P=[2,5,6] , Ci sono match parziali di P: un match di lunghezza 1 che inizia col primo nodo di C, poi ci sono 2 match di lunghezza 2 che iniziano nella posizione 1 e nella posizione 3. Infine un match di lunghezza 1 che inizia nella posizione 5. Quindi la funzione match dovrebbe restituire in inizio il puntatore al secondo nodo di C e lung=2. Se invece i valori puntati da C fossero: 2 2 5 2 5 6 6 e P=[2,5,6], allora il match più lungo ha lunghezza 3 (è completo) e inizia nel nodo in posizione 3.

Correttezza:

1) Scrivere la POST di match
2) Dimostrare che la funzione buildT è corretta rispetto alle PRE e POST date.