preparazione II compitino

Programmazione ricorsiva: Data una lista concatenata L, i suoi nodi hanno posizione 0,1,2, e così via. Il tipo dei nodi di L è la solita struttura nodo che trovate nel file esercizio2.cpp.

Con L_(k1,k2), se 0<=k1<=k2 denoteremo la porzione di L dal nodo k1 al nodo k2 (compresi). Se k2<k1, allora L_(k1,k2)=0. Anche se k1 è maggiore della posizione U dell'ultimo nodo di L, allora L_(k1,k2)=0 e se k1<=U <k2, allora L_(k1,k2) = L_(k1,U).

Si richiede di scrivere una funzione ricorsiva
nodo* cut(nodo*&L, int k1, int k2)

che sia corretta rispetto alle seguenti PRE e POST:

PRE_cut=(L è una lista corretta, k1 e k2 sono definite e 0<=k1<=k2)

POST_cut=(F restituisce L_(k1,k2) attraverso il parametro L passato per riferimento e restituisce L_(0,k1-1)@L_(k2+1,U) col return).

Con U si indica la posizione dell'ultimo nodo di L. Nella POST_cut viene usato l'operatore @ che indica la concatenazione tra liste.

Esempio: *sia L=2->3->2->4->0->5 e k1=0 e k2=2. Allora L_(0,2)=2->3->2, L_(0,-1)=0 e L_(3,5)=4->0->5. Si noti che 5 in questo esempio è la posizione dell'ultimo nodo che in POST_F abbiamo indicato con U. Quindi F dovrebbe restituire L_(0,2)=2->3->2 attraverso L e L_(0,-1)@L_(3,5)= 4->0->5, col return. Se invece k1=3 e k2=3, allora L_(3,3)=4 è quello che F restituisce attraverso L, mentre L_(0,2)@L_(4,5)=2->3->2->0->5 è la lista restituita da F col return. Se invece k1=0 e k2=10,allora F deve restituire 0 col return e tutto L attraverso L. Infine, se k1=6 e k2=7, allora F restituisce L col return e 0 attraverso L.*

idea:

1) individuazione dell'inizio di L_(k1,k2)

➔scorrere la lista diminuendo sia k1 che k2 finché k1=0

e se finisce? Beh allora niente cambia

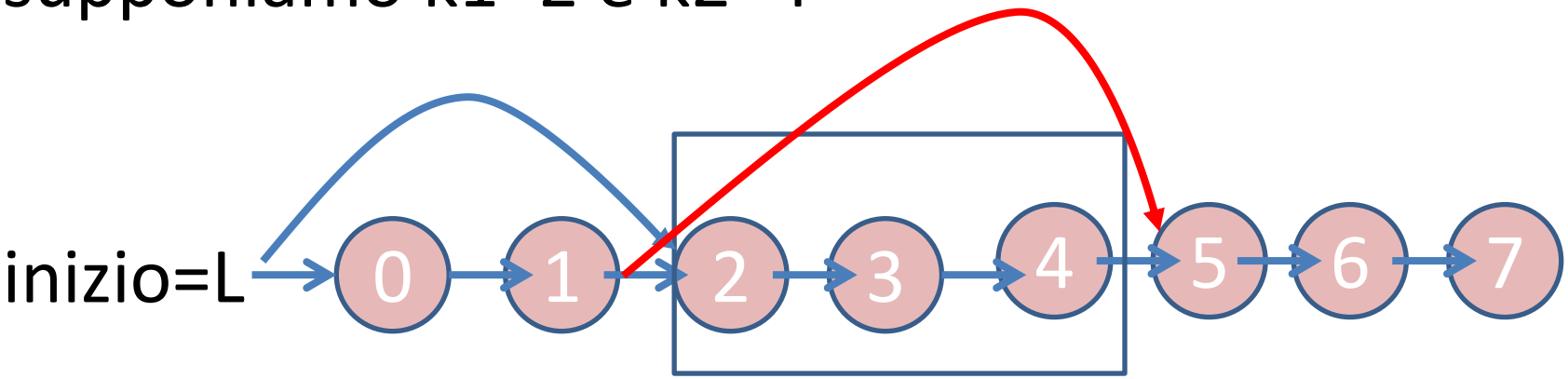
2) individuare la fine di L_(k1,k2)

➔ continuare a scorrere la lista diminuendo k2 fino a che anche k2 diventa 0

anche qui L può finire...è caso L_(k1,U) + semplice

ma dobbiamo staccare i nodi attraversati nella fase (1) e collegarli a quello che resta dopo (2)

supponiamo k1=2 e k2=4



(1)	k1	2	1	0		
	k2	4	3	2		
(2)	k2				1	0

in (1) devo staccare L dai nodi 0 e 1 chevanno restituiti col return e poi assegno 2 a L

quando finisce (2) devo chiudere 4 e ritornare 5,6 e 7 col return in modo che si attacchi a 0,1

```
nodo* cut(nodo*&L, int k1, int k2)
{ if(!L) return 0;
   if(k1) // fase (1)
    {nodo*x =L;
     L=L->next;  //stacco il nodo  e porto L nella ric.
     x->next=cut(L,k1-1,k2-1);
     return x;
    }
     else //fase(2)
     {  if(k2)
           return cut(L->next,k1,k2-1);
         else
           {nodo*x=L->next; //x è il resto
            L->next=0; //chiudo la lista
             return x; }
}
```

dimostrazione induttiva:
casi base:
!L) non c'è lista e quindi L_(k1,k2)=0, restituisco due 0, con L e col return => POST_cut

k1=k2=0) L_(0,0)= primo nodo di L, quindi restituisco quel nodo (dopo aver messo a zero il suo campo next) e col return restituisco L->next

passo induttivo:

k1>0) stacco il primo nodo (x) e porto avanti L, poi a x->next assegno il risultato di cut(L,k-1,k2-1) che per ipotesi induttiva restituisce L_(0,k1-1-1)@L(k2,U) col return e L_(k1-1,k2-1) con L

naturalmente devo dimostrare che vale PRE_cut_ric
PRE_cut=(L è una lista corretta, k1 e k2 sono definite e 0<=k1<=k2
POST_cut=(F restituisce L_(k1,k2) attraverso il parametro L passato per riferimento e restituisce L_(0,k1-1)@L_(k2+1,U) col return).

k1=0 e k2>0) devo restituire L_(0,k2) con L e L_(k2+1,U) col return
l'invocazione ricorsiva restituisce L_(0,k2-1) in L->next e L->next_(k2,U) col return
basta osservare che L_(k2+1,U)=L->next_(k2,U) visto che L->next ha il primo nodo in meno

k1=0 e k2=0) devo restituire con L L_(0,0), cioè il primo nodo, mentre L_(1,U) lo restituisco col return

Attenzione che metto a 0 il campo next del primo nodo

Dotare la seguente funzione ricorsiva di adeguate pre- e post-condizioni:

```
int F(nodo*R)
{ if(!R->left && !R->right)  return 1;
   if(!R->left) return F(R->right);
   if(!R->right) return F(R->left);
   return F(R->left)+F(R->right);
}
```

esercizio 1 di questa settimana:
(F1) eliminare gli ultimi k nodi con campo info1=y se ce ne sono almeno k in tutto
(F2) eliminare i primi k nodi se ce ne sono almeno k in tutto

nodo* F1(nodo*&L, int y, int k, int & v) e lo stesso per F2

Idea per F1 : all'andata usare v per contare il n. totale di nodi con info1=y

i) alla fine della lista se v>=k, si può rimettere v a 0 e ritornare su aumentando v ad ogni eliminazione fino a che v=k

se invece alla fine della lista v<k allora non si deve cambiare la lista e basta mettere v=k, per (i) risalendo nella ricorsione questo valore previene qualsiasi cambiamento nella lista.

Idea per F2 è simile, ma anche + facile:
all'andata v serve a contare i nodi con info1=y, come prima, ma i primi k nodi con info1=y già "sanno" di essere i nodi eventualmente chiamati ad essere staccati da L

alla fine della ricorsione, se v>=k basta inviare un "messaggio" verso l'alto per informare i nodi che "sanno" che devono staccare o no il loro nodo, questo lo si può fare con v, per esempio basta fare v=1 se devono staccare e v=0 se non devono staccare

nota: non importa arrivare in fondo alla lista