IV Compitino del 17/5/2016

Teoria: (5 punti)

1) spiegare se il seguente programma è corretto o no. La risposta va motivata brevemente, anche con l'aiuto di un grafico.

```
int ** F(int*&a){int ** b=&a;  (*b)++;  return b;}

main(){int A[]={0,1,2,3}, *q=A+1, **p=&q; *(F(*p))=A+2;  cout<<*q<<**p<<endl; }
```

2) Dato char X[3][4][5][6] e assumendo che X abbia R-valore X, che valore e che tipo ha l'espressione

(*(X[-2]))[2] ?

Programmazione: (19 punti)

Si tratta di leggere 3 interi, n_el, k e z, e poi di leggere n_el interi e contemporaneamente di costruire ricorsivamente una lista L(Q) di n_el nodi che contengono nel loro campo info gli interi letti. Abbiamo visto come fare questa operazione, tramite la funzione F dell'esercizio 0 dell'11/5/2016. Ricordiamo che, se Q punta al primo nodo di una lista concatenata, con L(Q) si denota l'intera lista.

Successivamente si vogliono rimuovere gli ultimi k nodi di L(Q) che hanno campo info=z. Questo va fatto però solamente se L(Q) contiene almeno k nodi. Altrimenti non va fatto nulla.

Esempio: se n_el=10, k=3 e z=0, e la lista L(Q) di 10 nodi fosse: 0-> 1-> 2-> 3-> 0-> 0-> 1-> 0-> 1-> 1, allora la funzione da fare dovrebbe restituire col return la lista degli ultimi 3 nodi con info=0 e rimarrebbe la lista L(Q) : 0-> 1-> 2-> 3-> 1-> 1-> 1.

Se n_el=10, k= 4 e z=0 e la lista Q fosse: 0-> 1-> 2-> 3-> 0-> 0-> 1-> 0-> 1-> 1, allora col return andrebbe restituita la lista che consiste di tutti e 4 i nodi con info=0, quindi 0->0->0->0, mentre L(Q) diventerebbe: 1-> 2-> 3-> 1-> 1-> 1

Se n_el=10, k= 0 e z=0 e la lista L(Q) fosse: 0-> 1-> 2-> 3-> 0-> 0-> 1-> 0-> 1-> 1, allora col return andrebbe restituita la lista vuota mentre L(Q) resterebbe inalterata. Infatti k=0 richiede di non rimuovere alcun nodo da L(Q).

Infine, se n_el=10, k= 5 e z=0 e la lista L(Q) fosse: 0-> 1-> 2-> 3-> 0-> 0-> 1-> 0-> 1-> 1, poiché L(Q) non contiene 5 nodi con campo info=0 (ma solo 4), L(Q) deve restare inalterata e si deve restituire 0 col return. Nessun nodo viene rimosso da L(Q).

La funzione che esegue queste operazioni è **ricorsiva** e deve essere conforme alla seguente specifica:

PRE=(L(Q) è lista corretta, k>=0, z def., ck def, vL(Q)=L(Q), vk=k e vck=ck)

```
nodo* sUk(nodo*&Q, int k, int z, int & ck)  //sUk  sta per stacca ultimi k (nodi)
```

POST=(se vL(Q) contiene x nodi con info=z e vck+x >=k, allora viene restituito col return la lista che consiste degli ultimi k-ck nodi di vL(Q) che contengono info=z ed L(Q) è vL(Q) meno i nodi restituiti col return) &&(se vL(Q) contiene x nodi con info=z e vck+x<k, allora L(Q)=vL(Q) e viene restituito 0 col return)

MOLTO Importante: Si osservi con attenzione la POST. Infatti essa spiega come usare il parametro ck (passato per riferimento). All'andata, ck conta i nodi con info=z in modo da sapere se ce ne sono almeno k. Al ritorno invece indica quanti nodi con info=z devono ancora venire rimossi percorrendo la lista L(Q) a ritroso. Quindi k-ck è il numero dei nodi che sono già stati rimossi.

Attenzione: viene fortemente sconsigliato l'uso di una funzione che conta quanti nodi con info=z sono presenti in L(Q). Nessun nuovo nodo deve essere creato, né alcun nodo va deallocato. Si maneggiano solo i nodi di L(Q) iniziale.

Correttezza (8 punti): scrivere la prova di correttezza induttiva di sUk rispetto a PRE e POST.