Esercizi del 21/2/2013

INFORMAZIONI IMPORTANTI: Il programma da fare deve risiedere sul file esercizio1.cpp e deve leggere l'input da input1 e scrivere l'output su output1. Il comando di consegna è consegna esercitazione.

Teoria: Il seguente programma è corretto o no? Spiegate la vostra risposta e, se pensate sia corretto, spiegate cosa stampa e perché:

```
int & g(int ** x){int *p=*x+1; *p=**x+*p; return *(p-1);}
main() {int X[]={1,2,3}, *q=X+1; g(&q)=X[1]; cout<<X[0]<<X[1]<<X[2]<<endl;}
```

Programmazione: dato un array X di 10 interi, si tratta di leggere interi da input1, continuando fino a che non si legga la sentinella -2 e, dopo averla letta, si deve distinguere tra due situazioni:

i) prima di leggere la sentinella -2 non sono stati letti più di 10 interi;

ii) prima di leggere la sentinella -2 sono stati letti più di 10 interi.

Nel caso (i) si devono stampare tutti i valori letti (prima della sentinella) nello stesso ordine in cui sono stati letti. Nel caso (ii) si devono stampare gli ultimi 10 valori letti (prima della sentinella) nell'ordine in cui sono stati letti. Per eseguire questo compito si deve usare SOLAMENTE l'array X di 10 elementi.

Nel caso (i), X è ovviamente sufficiente a contenere i valori letti prima della sentinella. Quindi nulla di sorprendente. Invece nel caso (ii), le 10 posizioni di X sono sufficienti a contenere gli ultimi 10 valori letti. Il problema è che man mano che si leggono i valori, non si sa quali siano gli ultimi 10 e quindi, dopo aver letto in X i primi 10 valori, l'11-esimo va messo al posto del primo letto (che sicuramente non è tra gli ultimi 10 letti e quindi può venire dimenticato), il 12-esimo letto va messo al posto del secondo (per lo stesso motivo di prima) e così via. Insomma X conterrà in ogni momento gli ultimi 10 letti in quel momento, ma in generale non li conterrà a partire dalla prima posizione in poi. La gestione di X è in un certo senso "circolare" come illustra il seguente esempio.

Esempio: *Per semplicità supponiamo che X abbia 4 posizioni (anziché 10). Consideriamo innanzitutto il caso (i). Supponiamo che input1 contenga 1 2 -2. Allora il programma inserisce in X i due interi che precedono -2 cioè M sarà [1,2,_,_] e poi, dopo aver letto -2, stamperà 1 e 2. Fine.*

Consideriamo ora il caso (ii). Supponiamo che input1 contenga [3, 1, 2, 4, 0,3,-2]. Allora il programma da realizzare legge uno alla volta i valori inserendoli in X (che ha ha 4 elementi), quindi dopo 4 letture X diventa: [3,1,2,4], il quinto valore viene letto nella prima posizione e quindi X=[0,1,2,4] e con la sesta lettura X=[0,3,2,4]. Questo X contiene gli ultimi 4 valori letti, ma per stamparli si deve partire dalla terza posizione, poi la quarta,e infine le prime 2. Il problema consiste esattamente nello scrivere una funzione capace di realizzare questa gestione "circolare" di X. Infatti la soluzione richiesta non può fare altre operazioni su X oltre ad inserirvi i valori letti. Non sono ammessi spostamenti di valori dentro a X.

In pratica si tratta di scrivere una funzione G(int X[], ifstream & INP, ofstream & OUT); che soddisfi le seguenti pre e post-condizioni.

PRE_F=(X ha 10 posizioni) && (vale caso (i) o caso (ii)), specificati di seguito:

caso (i)= (input1 contiene w -2…, con w = seq. di non più di 10 valori)

caso (ii)=(input1 contiene w1 w2 -2 .., con w2 = seq. di 10 int e w1= seq. non vuota di interi)

POST_F=(nel caso (i) output1 contiene w) &&(nel caso (ii) output1 contiene w2)

NB: nell'output ogni valore intero deve essere seguito da almeno uno spazio.

Oltre alla funzione F si deve produrre un main che apra i file e invochi F.

Programmazione 2: Se avanza tempo provate questo esercizio facile e formativo. Si tratta di scrivere una funzione void SEL(int X[], int & dim, int k) che soddisfa le seguenti pre e post:

PRE_SEL=(X ha dim elementi definiti e k è definito)&&(X=vX) &&(vdim=dim) vX e vdim sta per vecchio a indicare i valori originali di X e dim

POST_SEL=(X contiene esattamente tutti i valori di vX diversi da k e nello stesso ordine relativo che avevano in vX) &&(il numero di elementi che restano in X è dim<=vdim)