Scritto di Programmazione 21/3/2013

Informazioni importanti: trovate nella vostra home 2 file esercizio1.cpp ed esercizio2.cpp. Il primo concerne la domanda iterativa ed il secondo quella ricorsiva. A ciascun file dovete aggiungere le funzioni richieste dalla domanda corrispondente. I 2 file contengono il main che esegue l'apertura dei file e le letture e scritture da e sui file. Inoltre ciascun main invoca le funzioni che sono da sviluppare per l'esame. Il comando di consegna è, come sempre, "consegna esercitazione". Le parti che riguardano la correttezza vanno aggiunte in fondo al file come commento.

Programmazione iterativa: abbiamo un array int X[lim1][5][10], lim1>0, che contiene dim>0 elementi definiti e due interi k1 e k2>=0. Chiamiamo fette di X le sequenze di righe di strati sovrapposti. Quindi la fetta 0 sarà la sequenza delle righe 0 degli strati di X, la fetta 1 la sequenza delle righe 1 e così via. Ovviamente, come l'intero array, anche le sue fette saranno in genere riempite solo parzialmente di elementi definiti. Si tratta di scrivere una funzione F che restituisca l'indice di una fetta di X che soddisfi la seguente condizione (*): la fetta contiene k2 volte il valore k1.

Ovviamente nel verificare questa condizione vanno considerati solo gli elementi definiti della fetta.

Esempio: *Supponiamo per semplicità che X[2][3][4] e che dim=17, questo significa che c'è un solo strato completamente definito e che il secondo ha la prima riga completamente piena, la seconda con 1 solo valore definito e la terza completamente indefinita. Quindi le fette saranno così: la fetta 0 sarà completamente definita (2 righe di 4 elementi ciascuna), la fetta 1 avrà 1 riga completa di 4 elementi definiti ed una seconda riga con 1 solo elemento definito. La fetta 2 è composta solo di 1 riga completamente definita (4 elementi).*

Una volta che si è capito quali sono gli elementi definiti delle fette, scrivere un codice che calcoli se una fetta soddisfa (*) non dovrebbe essere difficile. La pre- e postcondizione di F ed il suo prototipo sono come segue:

PRE_F=(X ha dimensioni [][5][10], lim1>0, dim>0, i primi dim elementi di X sono definiti, k1 e k2 sono definiti e k2 >=0)

int F(int(*X)[5][10], int lim1, int dim, int k1, int k2)

POST_F=(F restituisce il minimo indice tra le fette che soddisfano (*) , se nessuna fetta soddisfa (*) allora F restituisce -1)

F dovrebbe eseguire il calcolo richiesto usando almeno una funzione iterativa ausiliaria che, dato un qualsiasi array a 1 dimensione ed un valore k1, restituisca il numero di occorrenze di k1 nell'array. In fondo ogni fetta è composta da un certo numero di righe di lunghezza eventualmente diversa. Ogni funzione ausiliaria che venga introdotta deve essere accompagnata da pre- e postcondizioni.

E' richiesto di specificare l'invariante del ciclo principale della funzione F.

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.*

E' richiesta la dimostrazione induttiva della correttezza di cut rispetto a PRE_cut e POST-cut.