Scritto di Programmazione del 18/1/2018

In questo esercizio usiamo la struttura FIFO che è come segue:

struct FIFO{nodo*primo,*ultimo; int dim;}; //attenzione al campo dim che servirà nell'esercizo.

e anche la struttura nodoFIFO come segue:

struct nodoFIFO{FIFO info; nodoFIFO*next;};

La struttura nodo è la solita usata per le liste concatenate.

Il problema consiste nel leggere 2 interi n e m, entrambi maggiori di 0, e poi di leggere n interi e di costruire una lista Q di nodi di tipo nodoFIFO che rispetti le seguenti proprietà:

- La struttura FIFO contenuta in ogni nodo di Q gestisce una lista ordinata di lunghezza uguale a m, tranne quella dell'ultimo nodo di Q che può essere più corta di m (ma non vuota).
- Se concatenassimo tra loro tutte le liste gestite dai FIFO dei nodi di Q (dal primo all'ultimo nodo di Q), otterremmo una lista di n nodi, ordinata in modo crescente e contenente gli n valori letti da cin.

Un esempio chiarirà meglio quello che si vuole.

Esempio. Sia n=5 e m=2 e supponiamo che i 5 valori che seguono siano 3 1 5 2 4. Inizialmente Q=0. Dopo la lettura di 3, Q dovrà avere un nodo con una FIFO che gestisce la lista con un nodo con info=3 e dim=1. Dopo la lettura di 1, l'unico nodo di Q dovrà contenere una FIFO con 1->3 e dim=2. Il terzo valore, 5, non può trovare posto nel primo e unico nodo di Q perché m=2 è il massimo numero di nodi consentito per ciascun FIFO. Quindi, dovrà venire creato un secondo nodo di Q con una FIFO che gestisce la lista con 5 e dim=1. Si osservi che, sebbene adesso abbiamo 2 FIFO separate, complessivamente i loro nodi sono ordinati rispetto al campo info. Questo dovrà sempre essere garantito dopo ogni lettura. Veniamo al quarto valore, 2. Per incorporarlo in Q, mantenendo l'ordine, il primo nodo di Q dovrà contenere una FIFO 1->2, mentre il secondo nodo avrà una FIFO con 3->5 e dim=2. Per incorporare l'ultimo valore, 4, si dovrà creare un terzo nodo di Q e i 3 nodi avranno FIFO che gestiscono, rispettivamente: 1->2, 3->4 e infine il nodo 5 da solo. Si osservi che il nodo 3, inizialmente gestito dal primo nodo di Q, alla fine è gestito dal secondo nodo di Q. Il programma deve essere in grado di effettuare tali spostamenti.

Esercizio. Per effettuare le operazioni descritte, si chiede di sviluppare 2 funzioni iterative ed una ricorsiva come segue. Le funzioni iterative sono le seguenti:

PRE=(x gestisce una lista non vuota ordinata in modo crescente)

FIFO addord(FIFO x, nodo*N)

POST=(restituisce una FIFO che gestisce una lista che è quella gestita da x a cui è stato aggiunto il nodo N mantenendo l'ordinamento, i campi primo, ultimo e dim del valore FIFO restituito devono essere corretti rispetto alla lista che gestisce, in particolare, il campo dim deve essere incrementato di 1 rispetto a x.dim iniziale)

PRE=(x gestisce una lista non vuota)

nodo* poplast(FIFO& x)

POST=(toglie dalla lista gestita da x l'ultimo nodo e lo restituisce col return, alla fine x deve essere un valore FIFO corretto (anche dim) rispetto alla lista che resta (possibilmente vuota))

La funzione ricorsiva è come segue:

PRE=(Q è una lista di nodoFIFO corretta (potrebbe essere anche vuota), N è un nodo allocato con next=0, m>0)

nodoFIFO* addtolist(nodoFIFO*Q, nodo*N, int m)

POST=(restituisce una lista Q' di nodoFIFO ottenuta da Q inserendo in Q il nodo N in modo tale da mantenere il fatto che complessivamente le liste gestite dai FIFO dei nodi di Q' sono ordinate e sono tutte di lunghezza m, a parte l'ultima che ha lunghezza tra 1 e m)

La funzione addtolist fa uso delle 2 funzioni iterative per raggiungere il suo scopo.

Attenzione: Il programma deve creare n nodi di tipo nodo, uno per ogni intero letto e un numero di nodi nodoFIFO sufficiente per costruire la lista Q richiesta. Nessun altro nodo deve venire creato e nessun nodo deve venire deallocato.

Correttezza:

1) Specificare l'invariante e la postcondizione del ciclo della funzione addord.
2) Per il ciclo di (1), dimostrare che l'invariante, assieme alla negazione della condizione di permanenza implica la postcondizione del ciclo.
3) Sempre per addord, dimostrare infine che la postcondizione e le istruzioni che seguono il ciclo della domanda (1), dimostrano la POST della funzione.