

Scritto di programmazione del 28/6/2016 Turno 1

Teoria (3 punti) Dire se il seguente programma è corretto o no e spiegare la risposta.

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

```
main() {int A[]={0,1,2,3,4},*p=A+2, **q=&p, *&y= *F(q); y++; cout<<*p<<' '<<*y<<endl;}
```

Programmazione: ci sono 3 domande di programmazione di difficoltà crescente.

1) (4 punti) La prima riguarda la struttura dati FIFO che serve a gestire una lista concatenata attraverso due campi: primo e ultimo che puntano al primo e all'ultimo nodo della lista gestita. Per la struttura FIFO nel programma dato trovate la funzione push_end che dato un valore FIFO X ed un nodo aggiunge il nodo in fondo alla lista gestita da X. Viene chiesto di costruire una funzione iterativa FIFO concF(FIFO a, FIFO b) che costruisca una lista che concatena le liste gestite da a e b e restituisca un valore FIFO che gestisca la lista prodotta.

2) (12 punti) La seconda domanda chiede di definire una funzione **ricorsiva** che, dato un albero binario, e un valore $k > 0$, produce una nuova lista concatenata tale che i nodi di questa lista hanno i campi info uguali a quelli dei nodi dell'albero che si trovano nelle posizioni $k, 2*k, 3*k$, eccetera, relativamente all'ordine dei nodi dell'albero determinato dall'attraversamento infisso dell'albero. D'ora in poi chiameremo questo ordine dei nodi semplicemente **l'ordine infisso**. La funzione deve usare la struttura FIFO e la funzione concF del punto (1) e infatti ha prototipo:

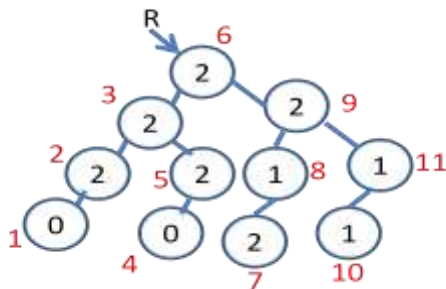
FIFO pickric_infix(nodot*R, int &n, int k) La PRE e POST seguono. La POST cerca di spiegare la funzione del parametro n (passato per riferimento). L'esempio successivo è forse più comprensibile.

PRE=(Tree(R) corretto, $1 \leq n \leq k$, k definito, sia $v_n = n$)

POST=(restituisce un valore FIFO f tale che f.primo sia una lista tale che i nodi abbiano i campi info di valore uguale ai campi info dei nodi dell'albero Tree(R) che si trovano nelle seguenti posizioni rispetto all'ordine infisso: sia $w = (k - v_n) + 1$, allora le posizioni da considerare sono $w, w+k, w+2*k, w+3*k$, eccetera. Inoltre $n-1$ è il numero di nodi presenti nell'albero che seguono, rispetto all'ordine infisso, il nodo dell'albero in corrispondenza del quale si è creato il nodo f.ultimo)

Esempio

dato l'albero



in rosso di fianco ad ogni nodo mettiamo la sua posizione nell'ordine infisso. Si osservi che le posizioni iniziano da 1. Quindi, se $k=2$ e $v_n=1$, la lista da costruire sarà: $2 \rightarrow 0 \rightarrow 2 \rightarrow 1 \rightarrow 1$ che corrisponde ai nodi nelle posizioni 2, 4, 6, 8, e 10 dell'ordine infisso. Quando pickric_infix crea l'ultimo nodo della lista, chiamiamolo

a, il nodo 10 sarà il nodo corrente e l'invocazione ricorsiva deve ritornare al nodo 11 con $n=1$ che indica che 11 è il prossimo nodo incontrato nell'ordine infisso dopo aver creato a. Quando `pickric_infix` lascia il nodo 11 per tornare alla radice R, n deve essere 2 come richiede la POST: $2-1=1$ e infatti un solo nodo nell'ordinamento infisso è stato attraversato dopo aver creato a. Inoltre $n=2$ dice che il prossimo nodo che verrà attraversato secondo l'ordine infisso dopo la radice R (dobbiamo immaginare che R sia un nodo interno di un albero più grande) sarà il secondo nodo nell'ordine infisso dopo aver creato a e, visto che $k=2$, dovremo usare il suo campo `info` per costruire un nuovo nodo da aggiungere in fondo alla lista che stiamo costruendo. Insomma n è il contatore che ci dice quando dovremo creare il prossimo nodo e appenderlo in fondo alla lista che stiamo costruendo.

Se invece $vn=2$ e $k=2$, allora i nodi dell'albero da considerare sarebbero $(2-2)+1=1$, cioè il nodo 1 nell'ordine infisso, che ha `info=0`, poi il nodo $1+2=3$ con `info=2`, poi il nodo $1+2*2=5$ con `info=2`, e ancora il nodo $1+3*2=7$ con `info=2`, poi il nodo $1+4*2=9$ con `info=2` e infine il nodo $1+4*2=11$ con `info=1`. Quindi la lista da costruire sarebbe: $0 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 1$. Alla fine della funzione n sarebbe 1 a indicare che l'ultimo nodo considerato è anche l'ultimo presente nell'albero rispetto all'ordine infisso.

Con lo stesso albero consideriamo che succede con $k=3$ e $vn=1$: la lista corrisponderà ai nodi 3, 6, 9 dell'albero e quindi sarà: $2 \rightarrow 2 \rightarrow 2$, restano 2 nodi nell'albero rispetto all'ordine infisso (il 10 e l'11) e quindi alla fine $n=3$, come prescritto dalla POST.

Se $k=4$ ed $vn=3$, allora la lista corrisponderà ai nodi $4-3+1=2$, e poi $2+4=6$ e infine $2+2*4=10$ e quindi la lista da costruire è: $2 \rightarrow 2 \rightarrow 1$ mentre n alla fine sarà 2 visto che resta 1 solo nodo (l'11) nell'albero dopo il nodo 10 che è servito a generare il terzo nodo della lista secondo l'ordine infisso.

3) (8 punti) Il terzo esercizio richiede di scrivere una funzione **iterativa** che, data la lista prodotta dalla funzione `pickric_infix` dell'esercizio (2), elimina dalla lista stessa i nodi con campo `info` ripetuti lasciando un solo nodo per ogni valore di campo `info` presente. Il nodo che resta per ogni valore di campo `info` deve essere il primo con quel valore nella sequenza originale

Esempio: se la sequenza è: $Q=2 \rightarrow 0 \rightarrow 3 \rightarrow 0 \rightarrow 2 \rightarrow 1 \rightarrow 0$, allora si vuole ottenere la sequenza $2 \rightarrow 0 \rightarrow 3 \rightarrow 1$ dove vengono tolti dalla sequenza x il quarto, il quinto e l'ultimo nodo. I nodi tolti formano la lista $2 \rightarrow 0 \rightarrow 0$. Osservare che la lista dei nodi tolti è prodotta considerando i nodi di Q dall'inizio (iniziando quindi da 2) e inserendo nella lista dei nodi tolti gli altri nodi con lo stesso `info` che si trovano alla sua destra. Quindi partendo dal nodo iniziale con `info=2`, va eliminato il 2 in quinta posizione. Poi si considera il secondo nodo, con `info=0` e si eliminano gli altri 2 nodi con `info=0`, cioè il quarto e l'ultimo. La lista dei nodi tolti sarà quindi quella detta prima, $2 \rightarrow 0 \rightarrow 0$, mentre della lista iniziale resterà $2 \rightarrow 0 \rightarrow 3 \rightarrow 1$.

Aiuto: visto che si vuole scorrere iterativamente una lista per eliminare alcuni suoi nodi, conviene introdurre un array dinamico di puntatori ai nodi della lista. In questo modo ci si può muovere liberamente sui nodi della lista. Si osservi però che quando si eliminano i nodi della lista, ci saranno elementi dell'array che non puntano più a nulla. Si dovrà fare molta attenzione a gestire questi casi in modo appropriato. Si osservi che la costruzione dell'array di puntatori che viene consigliata, in un certo senso è una simulazione dello stack dei record di attivazione che verrebbe costruito durante l'esecuzione di una funzione ricorsiva che scorra la lista per mezzo della ricorsione.

La funzione da fare ha il seguente prototipo: `FIFO tieni_primo(nodo*& Q)` e deve rispettare le seguenti PRE e POST:

$PRE=(L(Q) \text{ è corretta}, vL(Q)=L(Q))$

POST=(L(Q) è vL(Q) in cui sono stati eliminati i nodi che hanno un campo info tale che un nodo alla loro sinistra abbia lo stesso campo info. Quindi per ogni valore info resta solo il primo nodo con quel valore. La lista dei nodi eliminati va gestita da un valore FIFO restituito col return)

Correttezza :

- 1) **(5 punti)** delineare la prova induttiva della correttezza di pickric_infix
- 2) **(3 punti)** scrivere l'invariante del ciclo principale di tieni_primo.