Scritto di programmazione del 28/6/2016 Turno 2

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.

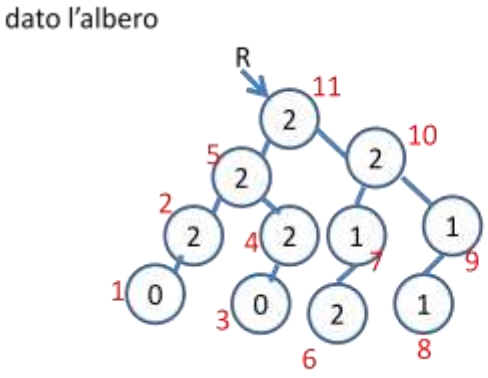
2) **(12 punti)** La seconda domanda chiede di definire una funzione **ricorsiva** che, dato un albero binario, e un valore k>0, produce una nuova lista concatenata tale che i nodi di questa lista hanno i campi info uguali a quelli dei nodi dell'albero che si trovano nelle posizioni k, 2*k, 3*k, eccetera,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

FIFO pickric_postfix(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<= n<=k, k definito, sia vn=n)

POST=(restituisce un valore FIFO f tale che L(f.primo) sia una lista L tale che i nodi abbiano i campi info di valore uguale ai campi info dei nodi dell'albero Tree(R) che si trovano nella posizione vn, vn+k, vn+2*k, eccetera rispetto all'ordine postfisso, inoltre n-1 è il numero di nodi presenti nell'albero che seguono, rispetto all'ordine postfisso, il nodo dell'albero in corrispondenza del quale si è creato l'ultimo nodo di L)

Esempio:



in rosso di fianco ad ogni nodo mettiamo la sua posizione nell'ordine postfisso,. Si osservi che le posizioni iniziano da 1. Quindi, se k=2 la lista da costruire sarà: 2->2->2->1->2 che corrisponde ai nodi nelle

posizioni 2, 4 , 6, 8, e 10 dell'ordine postfisso. Quando pickric_postfix 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 postfisso dopo aver creato a. Quando pickric_postfix lascia la radice 11, n deve essere 2 come richiede la POST: 2-1=1 e infatti un solo nodo nell'ordinamento postfisso (la radice 11) è stato attraversato nell'albero dopo aver creato a. Inoltre n=2 dice che il prossimo nodo che verrà attraversato secondo l'ordine postfisso dopo la radice (dobbiamo immaginare che R sia un nodo interno di un albero più grande) sarà il secondo nodo nell'ordine postfisso dopo aver creato a. Insomma n è il contatore che ci dice quando dovremo creare il prossimo nodo e appenderlo in fondo alla lista che stiamo costruendo.

Con lo stesso albero consideriamo che succede con k=3: la lista corrisponderà ai nodi 3, 6, 9 dell'albero e quindi sarà: 0->2->1, restano 2 nodi nell'albero rispetto all'ordine postfisso (il 10 e l'11) e quindi n=3 alla fine della funzione.

Se k=4, allora la lista corrisponderà ai nodi 4 e 8 e quindi sarà: 2->1 mentre n=4 visto che restano 3 nodi (9, 10 e 11) nell'albero secondo l'ordine postfisso.

3) **(8 punti)** Il terzo esercizio richiede di scrivere una funzione **iterativa** che, data la lista prodotta dalla funzione pickric_postfix 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->0->3->0->2->1->0, allora si vuole ottenere la sequenza 2->0->3->1 dove vengono tolti dalla sequenza x il quarto, il quinto e l'ultimo nodo. I nodi tolti formano la lista 2->0->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->0->0, mentre della lista iniziale resterà 2->0->3->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) 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 d pickric_postfix
2) **(3 punti)** scrivere l'invariante del ciclo principale di tieni_primo.