# Programmazione II– Canale P-Z
# Esame del 11/06/04

## Schema delle soluzioni

**Esercizio 1.** Data una lista di interi, definiamo *somma a segno alterno* degli elementi nella lista il valore che si ottiene sommando tutti gli elementi della lista in posizione dispari (il primo elemento è in posizione 1, il secondo in posizione 2, etc.) e sottraendo tutti quelli in posizione pari. Ad esempio, se la lista contiene i seguenti elementi

$$12, -23, 15, 21, -1, 27$$

nell'ordine in cui sono stati riportati, la sua somma a segno alterno sarà pari a

$$+12 - (-23) + 15 - 21 + (-1) - 27 = 1$$

Dopo aver dato le definizioni dei tipi necessari, si scrivano

1. una funzione iterativa che calcola la somma alterna di una lista;
2. una funzione ricorsiva che calcola la somma alterna di una lista.
   *Suggerimento*: Una possibile soluzione consiste nel capire che relazione c'è tra la somma alterna precedentemente definita e quella in cui si sommano gli elementi della lista in posizione pari e si sottraggono quelli in posizione dispari. Oppure, per trovare un'altra soluzione, si tenga conto che non è detto che in una ricorsione su lista, ad ogni passo, si utilizzi e si rimuova dalla lista per la successiva chiamata ricorsiva solo l'elemento di testa.

**Schema della soluzione** La soluzione ricorsiva più semplice per il calcolo di $f$ si basa sulla seguente osservazione: prendiamo una lista $l$ contenente nell'ordine i valori $l_0, l_1, \ldots, l_{2i}, l_{2i+1}, \ldots$, allora $f(l) = l_0 - l_1 + l_2 - \ldots + l_{2i} - l_{2i+1} + \ldots$ è pari a $l_0 - (l_1 - l_2 + \ldots - l_{2i} + l_{2i+1} - \ldots) = l_0 - f(l')$ dove $l'$ è la coda di $l$. È pertanto sufficente implementare la precedente funzione, tenendo conto che $f$ della lista vuota è 0.

Se invece si vuole dare una soluzione iterativa, la cosa più semplice da fare è mantenere una variabile booleana $b$ (ovvero, un intero da trattare come valore booleano) da inizializzare a true e da negare ogni volta che si accede all'elemento successivo della lista (garantendo così che $b$ è true se ci si trova su di un elemento in posizione pari, e false se ci si trova su di un elemento in posizione dispari). In questo modo, durante la scansione, si dovranno sommare gli elementi della lista acceduti con $b$ pari a true, e sottratti quelli acceduti con $b$ pari a false.

**Esercizio 2.** Utilizzando uno stack, scrivere una funzione iterativa che riceve come parametro una stringa contenente parentesi tonde o quadre e che risponde vero (un intero diverso da 0) se le parentesi nella stringa sono correttamente bilanciate e falso (il valore 0) se ci sono delle parentesi aperte che non sono state chiuse o se ci sono parentesi chiuse che non corrispondono ad una parentesi aperta dello stesso tipo. Prima di scrivere la funzione si forniscano anche i tipi di dato necessari.

**Schema della soluzione** Dopo aver correttamente inizializzato lo stack in base all'implementazione scelta, si può procedere nel seguente modo: si legge la successiva parentesi nella stringa, se è una parentesi aperta la si mette nello stack, se è una parentesi chiusa, si esegue una pop dallo stack e si verifica che la parentesi prelevata dallo stack è dello stesso tipo di quella chiusa.

La funzione deve ritornare

- true, se si arriva alla fine della stringa di parentesi con lo stack vuoto;
- false, nel caso in cui, al momento della lettura di una parentesi chiusa, lo stack è vuoto (la parentesi non era stata mai aperta) o si preleva dallo stack una parentesi del tipo errato (parentesi tonda/aperta chiusa da una di diverso tipo), oppure si arriva alla fine della stringa e lo stack non è vuoto (alcune parentesi aperte non sono state chiuse).

Si osservi che non va bene una soluzione basata su due indici che misurano il numero di parentesi aperte e non chiuse - uno per quelle tonde ed uno per quelle chiuse - da incrementare quando si trova una parentesi aperta del tipo corrispondente e da decrementare quando si trova una parentesi chiusa, verificando alla fine che si ottiene 0 senza eser mai passati per valori negativi degli indici. Ad esempio, verificare cosa risponderebbe una soluzione di questo tipo su ([)].

**Esercizio 3.** Si definisca un tipo di dato per la memorizzazione di alberi binari etichettati con interi e si scriva una funzione che rimpiazza l'etichetta di ogni nodo $v$ di un albero $T$ con la somma delle etichette dei nodi nel cammino che va dalla radice di $T$ al nodo $v$ (nella somma sono comprese le etichette della radice e del nodo $v$).

**Schema della soluzione** La soluzione più semplice consiste in una visita in preordine. Si deve cioè scrivere una funzione `rimpcamm` che riceve come parametri un nodo $v$ dell'albero e la somma $S$ delle etichette sul cammino dalla radice a $v$ escluso. Ad ogni passo, si dovrà aggiungere il valore dell'etichetta di $v$ a $S$, sostituire il valore $S'$ così ottenuto in $v$ e richiamare `rimpcamm` sui figli di $v$ con $S'$ come secondo parametro. Ovviamente, `rimpcamm` non dovrà fare nulla se $v$ è `NULL`, mentre, la prima chiamata ricorsiva sulla radice deve avvenire con 0 come secondo parametro.

**Esercizio 4.** Si scriva una funzione che ritorna un valore diverso da 0 se un albero verifica la seguente proprietà: in ogni nodo $v$ che non è una foglia dell'albero, la somma delle etichette nel sottoalbero sinistro di $v$ è minore dell'etichetta di $v$, mentre la somma delle etichette nel sottoalbero desto è maggiore di quella di $v$. Nel caso l'albero non verifica la precedente condizione la funzione dovrà ritornare 0. Si forniscano anche le definizioni dei tipi di dato utilizzati.
*Facoltativo.* Si discuta la complessità della funzione scritta. Se la soluzione proposta non è lineare, se ne cerchi una lineare.

**Schema della soluzione** Una soluzione lineare per il precedente algoritmo consiste nello scrivere una funzione che prende come parametro un nodo $v$ dell'albero e ritorna due valori, un valore $b$ vero/falso (diverso/ugale 0) a seconda che l'albero di radice $v$ verifica la proprietà e un intero $T$ pari alla somma dei nodi nell'albero $v$, che ha significato solo nel caso in cui $b$ è vero. Ad ogni nodo $v$ che non è una foglia, si verifica che i sottoalberi sinistro e destro verificano la proprietà, ottenendo al contempo la somma $T_s$ e $T_d$ dei sottolaberi sinistro e

destro di $v$. Quindi, se entrambi i sottoalberi verificano la proprietà, si confrontano $T_s$ e $T_d$ con l'etichetta $l$ di $v$ e, nel caso in cui la relazione richiesta non sia verificata, si termina la funzione ritornando false in $b$, altrimenti, si ritorna true in $b$ e $T_s + T_d + l$ in $T$.

Dato che una funzione C ritorna un solo valore, per ottenere due risultati occorre che uno o entrambi i valori di ritorno siano ottenuti mediante un parametro passato per riferimento. Ad esempio, $b$ ritornato come valore della funzione e $T$ restituito in un parametro passato per riferimento.