# Teoria

# Funzioni

* Implementare una funzione con firma:

ListNode\* alterna\_liste(ListNode \*l1, ListNode \*l2);

che, date due liste concatenate l1 e l2, restituisca una nuova lista contenente alternativamente gli elementi di l1 e l2. Se una lista è più lunga dell'altra, gli elementi rimanenti vanno aggiunti in coda alla lista risultante.

Esempio: l1: 1 -> 3 -> 5 l2: 2 -> 4 -> 6 -> 7 -> 8 Output: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8

* Scrivere una funzione ricorsiva con firma:

int cammino\_somma(int \*mat, int r, int c, int sum, int x, int y);

che, data una matrice mat di dimensioni r x c e un intero sum, determini se esiste un cammino dalla cella (0,0) alla cella (r-1, c-1) tale che la somma dei valori delle celle attraversate sia uguale a sum. È possibile muoversi solo in basso e a destra. La funzione deve restituire 1 se esiste un tale cammino, 0 altrimenti.

* Implementare una funzione con firma:

void elimina\_dispari(TreeNode \*\*root);

che, dato un albero binario di ricerca, elimini tutti i nodi contenenti valori dispari. L'albero risultante deve essere ancora un BST valido. Discutere la correttezza della funzione.

* Implementare una funzione con firma:

void zigzag\_stampa(TreeNode \*root);

che, dato un albero binario, stampi i suoi nodi in ordine "zig-zag": prima il livello 0 da sinistra a destra, poi il livello 1 da destra a sinistra, poi il livello 2 da sinistra a destra, e così via.

Esempio:

3

/ \

9 20

/ \

15 7

Output: 3 20 9 15 7

* Scrivere una funzione con firma:

int palazzo\_visibile(int \*altezze, int n);

che, dato un array altezze di n interi rappresentanti le altezze di n palazzi allineati, determini il numero di palazzi completamente visibili guardando dal lato sinistro della strada.

Un palazzo è completamente visibile se tutti i palazzi alla sua sinistra sono più bassi di esso.

Esempio:

Input: [3, 7, 8, 3, 6, 1]

Output: 3 (i palazzi di altezza 3, 7 e 8 sono completamente visibili)

* Implementare una funzione ricorsiva con firma:

int somma\_nodi(ListNode \*head, int k);

che, data una lista concatenata head e un intero k, restituisca la somma di tutti i nodi che si trovano a distanza k dalla fine della lista.

Esempio:

Input: 1 -> 2 -> 3 -> 4 -> 5 -> 6, k = 2

Output: 9 (4 + 5, i nodi a distanza 2 dalla fine)