# *Commento all'es. 1*

La versione ricorsiva base dell'esercizio è banale. Osservando che la condivisione dei sottoproblemi sembra essere rilevante, si propone, anche se non richiesto dalle specifiche, una versione basata su ricorsione con memoization.
Per calcolare fino a n numeri di Hofstadter, ricordando che essi sono per definizione > 0 e che i casi terminali sono Q(1) e Q(2), si alloca un vettore di dimensione n+1, la cui prima cella non sarà mai usata (non è definito Q(0)). Esso contiene le soluzioni già calcolate Q(i): se Q(i) è 0, allora bisogna calcolare ex novo la soluzione, se è diverso da 0, basta riusare quella soluzione.
Si osservi sperimentalmente come la soluzione con memoization sia molto più veloce della ricorsione standard.

# *Commento all'es. 2*

La versione ricorsiva base dell'esercizio è banale, in quanto si tratta solo di:

- nella condizione di terminazione (`n<b`) stampare il simbolo in corrispondenza di `n`
- nella discesa ricorsiva richiamare `converti` sul risultato della divisione intera di `n` per `b` e, una volta terminata la ricorsione, stampare il simbolo in corrispondenza di `n%b`.

Si osservi che non a caso si usa la terminologia "simbolo in corrispondenza di `n`". Le specifiche non pongono un limite superiore alla base `b`, quindi per basi `b>10` si pone la questione di quali siano i simboli leciti nella base. Visto che in base 16 i simboli che corrispondono ai decimali da 10 a 15 sono le lettere maiuscole da A ad F, per basi `b>10` si assumono come simboli le lettere dell'alfabeto inglese maiuscole e minuscole, realizzando quindi la possibilità di convertire fino a base 62 (10 + 26 + 26). Poiché `n` e `b` sono interi, come pure i risultati della loro divisione intera e il resto della divisione intera, si introduce una tabella di simboli che associa all'intero tra 0 e 61 il simbolo corrispondente e si usa questa tabella per la visualizzazione del risultato.

# *Commento all'es. 3*

La soluzione proposta è un'estensione di quella del laboratorio 4.1, in cui si utilizza lo stesso tipo di interfaccia utente modulare, basato su un doppio livello di opzioni (tipi `enum Comando` e `Criterio`). Non si utilizzano struct wrapper (ad eccezione della struct `ComandoCompleto`, usata per contenere tutte le informazioni relative a un comando e alle relative opzioni).

La struttura dati principale è un vettore di `Prodotto` (una struct), variabile locale del main.
I vettori ordinati (secondo vari criteri) di puntatori a `Prodotto` sono raccolti in un vettore di puntatori (`rifOrd`), a cui si accede tramite tipo `enum Criterio`: ad esempio, il vettore di puntatori ordinati per nome, si trova in `rifOrd[cr_nome]`. La funzione `leggiCatalogo` alloca i vettori e acquisisce i dati dal file, mentre la `generaOrdini` provvede a inizializzare e ordinare i puntatori (si noti il caso particolare del vettore `rifOrd[cr_nomeParziale]`, che fa riferimento a vettore di puntatori ordinati per nome completo).

Si noti infine come la funzione `confrontaProdotti`, a differenza del lab 4.1, riceva puntatori a `Prodotto`, anziché variabili di tipo `prodotto` passate per valore.