Esercitazione 3: Rubrica estendibile ordinata

Esercizio 1. Modificare la classe `Rubrica` vista a lezione, in modo che:

1. l'aggiunta di un elemento in una rubrica piena comporti, prima, il raddoppio della capacità della rubrica e, successivamente, l'effettiva aggiunta dell'elemento (anziché il fallimento dell'operazione).
2. i contatti vengano mantenuti in rubrica in ordine lessicografico crescente rispetto al campo *nome* del contatto;
3. tutti i metodi che necessitano di cercare un contatto tramite il *nome* sfruttino il fatto che i contatti sono mantenuti ordinati, per ridurre lo sforzo computazionale.

Scrivere un semplice programma di prova che verifichi il corretto funzionamento della classe.

Esercizio 2. Progettare e implementare una classe `Matrix` per rappresentare matrici $m \times n$ (con m righe e n colonne) di numeri interi. Dotare la classe di:

- metodi `get` e `set` che permettano di leggere/scrivere un elemento della matrice date le sue coordinate. Ad esempio, `m.get(0,0)` deve restituire l'elemento che si trova nell'angolo in alto a sinistra della matrice `m`;
- metodi `rows` e `columns` che ritornino rispettivamente il numero di righe e di colonne della matrice;
- un metodo `add` tale che `m1.add(m2)` ritorni la matrice ottenuta sommando `m1` ed `m2`;
- un metodo `mul` tale che `m1.mul(m2)` ritorni la matrice ottenuta moltiplicando `m1` con `m2`;
- un metodo `pow` tale che `m.pow(n)` ritorni la potenza `n`-esima della matrice `m`. Si ricorda che la potenza 0-esima di `m` è la matrice identità che ha lo stesso lato di `m`.

Specificare le pre-condizioni di ogni costruttore/metodo con una clausola `assert` opportuna.

Esercizio 3. Scrivere un metodo statico `mistero(n)` che ritorni l'elemento alle coordinate $(0, 0)$ della matrice

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

Che cosa calcola `mistero`?