

Simulazione di Prova Intermedia del 24/01/2023

(corrisponde alla seconda prova intermedia dell'AA 21/22)

Questa prova intermedia è costituita da due esercizi. Potranno essere consegnati al docente solo gli esercizi che avranno superato i nostri JUnit Test. I test svolti dagli studenti non saranno presi in considerazione per la valutazione. La consegna di almeno un esercizio è condizione necessaria, ma non sufficiente, per passare la prova.

Per la valutazione saranno presi in considerazione aspetti di “buona programmazione”, “pulizia del codice” ed efficienza. Ad es.: formattazione corretta del codice, rendere il codice modulare aggiungendo ove necessario altri metodi rispetto a quelli richiesti dall’esercizio, soprattutto se questi rendono il codice più pulito e leggibile, o se evitano duplicazione di codice. Inoltre, non ci devono essere warning nel codice scritto.

IMPORTANTE: seguire attentamente le specifiche per quanto riguarda i nomi dei metodi e la firma dei metodi, altrimenti i test automatici falliranno rendendo il compito insufficiente.

CONSEGNA ESERCIZI:

Entro il termine ultimo previsto per la consegna dello scritto, gli studenti che intendono consegnare provvedono a caricare il sorgente (o i sorgenti) **.javamm** attraverso l'apposita attività “compito”, disponibile nella pagina MOODLE del corso al seguente link:

<https://e-val.unifi.it/mod/assign/view.php?id=24981>

Fino allo scadere del tempo, lo studente potrà apportare modifiche al proprio lavoro. Non saranno accettate consegne effettuate in ritardo, o con modalità diverse da quelle definite dal docente. All'orario stabilito ad inizio compito il docente dichiara finita la prova e chiude la sessione.

Esercizio Java-- n. 1: Rimuovi Fibonacci da Intero

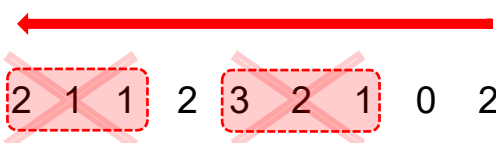
Sia n un numero intero (`int`), con $n \geq 0$. La “rimozione di Fibonacci” dal numero intero n è una procedura che genera un nuovo numero intero dato dalla rimozione da n di tutte le terne di tre cifre consecutive che sono legate fra loro dalla relazione di Fibonacci. Siano $c_m \dots c_1$ le m cifre che compongono il numero n , la procedura per la rimozione di Fibonacci è la seguente:

- Si parte dalla cifra meno significativa di n (c_1), e muovendosi verso le cifre più significative si ricerca una possibile terna di tre cifre consecutive (c_{k+2} , c_{k+1} , c_k) che sono legate fra loro dalla relazione di Fibonacci, ovvero tali che la somma delle prime due cifre consecutive è uguale alla terza ($c_{k+2} = c_{k+1} + c_k$, per un qualche $k \in \{1, \dots, m-2\}$).
- Una volta individuata la prima terna questa viene rimossa dal numero e quindi si prosegue nella ricerca della terna successiva a partire dalla cifra alla sinistra della terna rimossa (ovvero a partire da c_{k+3}).

NOTA:

- Nel caso in cui la procedura porti alla rimozione di tutte le cifre del numero n di partenza, il numero generato corrisponderà allo 0 (zero).
- Se il numero n contiene meno di 3 cifre, il numero generato corrisponderà allo stesso numero n .

Ad esempio, sia $n=211232102$. La rimozione di Fibonacci da n genera il numero intero 202, infatti (le cifre rimosse sono evidenziate in rosso):


 $n = 2\ 1\ 1\ 2\ 3\ 2\ 1\ 0\ 2 \rightarrow$ numero dopo la rimozione: 202

Infatti, partendo da destra le prime tre cifre consecutive per cui vale la relazione di Fibonacci sono (3 2 1) poiché $1+2=3$, e quindi le 3 cifre in rosso vengono rimosse. Si prosegue quindi dalla cifra successiva (2 nell'esempio), e la prossima terna da rimuovere è (2 1 1) (cifre in rosso).

Altri esempi:

- Se $n=0 \rightarrow$ numero dopo la rimozione = 0 (numero con meno di 3 cifre);
- Se $n=41 \rightarrow$ numero dopo la rimozione = 41 (numero con meno di 3 cifre);


- Se $n=514$, oppure $n=514211 \rightarrow$ numero dopo la rimozione = 0 (vengono rimosse tutte le cifre del numero);
- Se $n=51400211 \rightarrow$ numero dopo la rimozione = 0 (rimangono solo le due cifre 00 centrali);
- Se $n=53215321 \rightarrow$ numero dopo la rimozione = 55;

Scrivere un metodo Java--, chiamato **rimuoviFibonacci**, che dato in input un numero intero $n \geq 0$ restituisca il numero intero generato dalla rimozione di Fibonacci da n .

Estensione OPZIONALE dell'Esercizio Java-- n. 1:

Nello stesso file relativo all'esercizio n.1 (RimuoviFibonacciDaIntero.javamm), scrivere un metodo Java-- chiamato **rimuoviFibonacciEsteso**, che dato in input un intero $n \geq 0$, restituisca il numero intero generato dalla rimozione di Fibonacci da n e da tutti i numeri interi generati a partire da n fino ad ottenere un numero intero in cui non ci siano più cifre da rimuovere (quindi fino a quando il numero intero non può essere ulteriormente modificato).

Riprendendo l'esempio precedente in cui $n=211232102$, la rimozione *estesa* di Fibonacci genera il numero 4 (**rimuoviFibonacciEsteso**(n) = 0), infatti:

- **rimuoviFibonacci**(n) =  = $n1$;
- **rimuoviFibonacci**($n1$) = 0; \leftarrow è l'intero che deve restituire il metodo **rimuoviFibonacciEsteso**, in quanto il numero non contiene più cifre da rimuovere.

Nota bene:

- Saranno premiate le soluzioni che dichiareranno meno variabili di appoggio e occuperanno quindi meno memoria dati.
- I Junit Test da superare sono quelli della classe **RimuoviFibonacciTest** per il metodo di base, e quelli della classe **RimuoviFibonacciEstesoTest** per l'estensione opzionale.
- E' possibile invocare il metodo `Math.pow` per calcolare la potenza di un numero.
- Nello svolgere l'esercizio NON devono essere utilizzati i metodi `clone`, o `arraycopy`, o metodi della classe `Arrays`. L'utilizzo di tali metodi renderà l'esercizio automaticamente insufficiente.

Esercizio Java-- n. 2: Parola Nascosta

Sia T una matrice di caratteri (`char`) di dimensione $m \times n$, con $m > 0$ e $n > 0$, i cui caratteri corrispondono alle lettere maiuscole dell'alfabeto inglese {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}.

La lettera nascosta all'interno di una riga della matrice T corrisponde al carattere che compare più volte nella stessa riga della matrice (carattere con frequenza massima). Nota:

- Nel caso in cui ci siano più caratteri con la stessa frequenza massima, con frequenza massima ≥ 2 (ovvero con almeno 2 occorrenze del carattere), la lettera nascosta corrisponde al carattere con frequenza massima che compare per primo nella riga scorrendola da sinistra verso destra.
- Se tutti i caratteri della riga sono diversi fra loro (ovvero tutti i caratteri hanno frequenza uguale a 1), oppure se la riga è composta da un solo carattere, *non esiste nessuna lettera nascosta* all'interno della riga.

La parola nascosta nella matrice T corrisponde all'array di caratteri dato dalla sequenza di lettere nascoste trovate in ciascuna delle righe della matrice T , scorrendole dall'alto verso il basso. Nota:

- Nel caso in cui non esista alcun carattere nascosto in nessuna delle righe di T , la parola nascosta corrispondente sarà data da un array contenente 1 solo carattere uguale a ' ' (carattere di spazio – c'è uno spazio fra i due apici).

Ad esempio, sia $T = \begin{pmatrix} 'C' & 'U' & 'C' & 'C' & 'I' & 'A' \\ 'M' & 'E' & 'G' & 'L' & 'I' & 'O' \\ 'P' & 'A' & 'T' & 'A' & 'T' & 'E' \end{pmatrix}$:

- La lettera nascosta nella prima riga ('C' 'U' 'C' 'C' 'I' 'A') della matrice T è il carattere 'C', poiché è quello che compare il numero massimo di volte nella riga (3 volte);
- La lettera nascosta nella seconda riga ('M' 'E' 'G' 'L' 'I' 'O') della matrice T non esiste, poiché i caratteri della riga sono tutti diversi fra loro;
- La lettera nascosta nella terza riga ('P' 'A' 'T' 'A' 'T' 'E') della matrice T è il carattere 'A', poiché è quello che compare il numero

massimo di volte nella riga (2 volte, così come il carattere ' T ') e compare per primo nella riga scorrendola da sinistra verso destra.

- La parola nascosta nella matrice T è quindi data dall'array composto dai due caratteri: (' C ' ' A ').

Scrivere un metodo Java--, chiamato **cercaParolaNascosta**, che dato in input una matrice T come definita precedentemente, restituisca un array di caratteri corrispondente alla parola nascosta.

Nota:

- I Junit Test da superare sono quelli della classe **ParolaNascostaTest**.
- Nello svolgere l'esercizio NON devono essere utilizzati i metodi `clone`, o `arraycopy`, o metodi della classe `Arrays`. L'utilizzo di tali metodi renderà l'esercizio automaticamente insufficiente.