– Prima Prova Intermedia – 08/02/2022

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: (link rimosso)

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: Intero Nascosto

Siano *a*, *b*, *c* tre numeri interi (`int`) positivi, con *a*, *b*, *c* ≥ 0, tutti con lo stesso numero di cifre. La procedura per comporre il numero intero nascosto fra i tre numeri è la seguente:

- Si scorrono in modo allineato le cifre che compongono i tre numeri partendo da destra (dalla cifra meno significativa) verso sinistra (verso la cifra più significativa).
- Siano a_j, b_j, c_j le tre cifre in posizione *j* in ciascuno dei tre numeri. Se almeno due delle cifre a_j, b_j, c_j sono uguali fra loro allora tale cifra va a comporre il numero nascosto:

a =	a_m	…	a_j	…	a_0
b =	b_m	…	b_j	…	b_0
c =	c_m	…	c_j	…	c_0
	x_m	…	x_j	…	x_0

→ intero nascosto = $x_m \ldots x_j \ldots x_0$

Ciascuna cifra x_j è presente nel numero intero nascosto SOLO se almeno due delle cifre a_j, b_j, c_j sono uguali fra loro (ed il suo valore corrisponde proprio alla cifra uguale), altrimenti NON è presente.
Caso particolare: nel caso in cui tutte le cifre in ogni posizione *j* siano diverse fra loro, il numero intero nascosto corrispondente (che è privo di cifre) è il numero intero 0 (zero).

Ad esempio, siano *a*=937829, *b*=913803, *c*=212801, il numero intero nascosto sarà *n*=9180 poiché:

a =	9	3	7	8	2	9
b =	9	1	3	8	0	3
c =	2	1	2	8	0	1
	9	1		8	0	

→ intero nascosto = 9180

(le cifre uguali che vanno a formare il numero nascosto sono evidenziate in rosso)

Scrivere un metodo Java--, chiamato **interoNascosto**, che dati in input tre numeri *a*, *b*, *c* di tipo int (con *a*, *b*, *c* ≥ 0) restituisca il numero nascosto calcolato come precedentemente descritto.

Nota bene: E' possibile invocare il metodo Math.pow per calcolare la potenza di un numero. Saranno premiate le soluzioni che dichiareranno meno variabili di appoggio e occuperanno quindi meno memoria dati. I Junit Test che devono essere superati sono quelli della classe **InteroNascostoTest**.

Esercizio Java-- n. 2: Shuffle di Matrici

Sia T una matrice quadrata di interi (`int`) di dimensione m x m, con $m>0$. Sia S una matrice di interi (`int`) di dimensione n x m, con $n>0$, in cui ciascuna riga rappresenta una possibile permutazione degli indici riga (o colonna) della matrice T. In altri termini, ciascuna riga $s_h = (s_{h,0}, \dots, s_{h,m-1})$ della matrice S (con h=0,…,n-1) è un array di m numeri interi a valori compresi nell'intervallo $[0; m-1]$ e tutti diversi fra loro.

Uno shuffle verticale della matrice T rispetto ad una riga s_h della matrice S, genera una nuova matrice N delle stesse dimensioni di T in cui la colonna di indice k della matrice T viene copiata nella colonna di indice $s_{h,k}$ della matrice N, per ogni indice $k \in [0; m-1]$.

Ad esempio, sia $T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ e sia s_h = (1,0,2), lo shuffle verticale di T genera la matrice:

$$T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \xrightarrow{\text{Shuff. Vert.}} \begin{pmatrix} 2 & 1 & 3 \\ 5 & 4 & 6 \\ 8 & 7 & 9 \end{pmatrix} = N$$

- colonna 0 di T → colonna 1 ($s_{h,0}$) di N
- colonna 1 di T → colonna 0 ($s_{h,1}$) di N
- colonna 2 di T → colonna 2 ($s_{h,2}$) di N

Uno shuffle orizzontale della matrice T rispetto ad una riga s_h della matrice S, genera una nuova matrice N delle stesse dimensioni di T in cui la riga di indice k della matrice T viene copiata nella riga di indice $s_{h,k}$ della matrice N, per ogni indice $k \in [0; m-1]$.

Ad esempio, sia $T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ e sia s_h = (1,0,2), lo shuffle orizzontale di T genera la matrice:

$$T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \xrightarrow{\text{Shuff. Oriz.}} \begin{pmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 7 & 8 & 9 \end{pmatrix} = N$$

- riga 0 di T → riga 1 ($s_{h,0}$) di N
- riga 1 di T → riga 0 ($s_{h,1}$) di N
- riga 2 di T → riga 2 ($s_{h,2}$) di N

Uno shuffle completo della matrice T rispetto alla matrice S genera una nuova matrice come segue:

- Si applicano in sequenza gli shuffle verticali della matrice T rispetto ad ogni riga k della matrice S, con k=0, …, n-1 (in questo ordine);
- Sulla matrice risultante, si applicano in sequenza gli shuffle orizzontali rispetto ad ogni riga k della matrice S, con k=0, …, n-1 (in questo ordine).

Esempio 1: lo shuffle completo applicato alla precedente matrice T rispetto ad una matrice S composta dalla sola riga (1,0,2), genera la matrice:

$$T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \xrightarrow{\text{Shuff. Vert.}} \begin{pmatrix} 2 & 1 & 3 \\ 5 & 4 & 6 \\ 8 & 7 & 9 \end{pmatrix} \xrightarrow{\text{Shuff. Oriz.}} \begin{pmatrix} 5 & 4 & 6 \\ 2 & 1 & 3 \\ 8 & 7 & 9 \end{pmatrix} = N \qquad \text{(Esempio 1)}$$

Esempio 2: lo shuffle completo applicato alla precedente matrice T rispetto ad una matrice $S = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 2 & 1 \end{pmatrix}$, genera la matrice:

$$T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \xrightarrow[\text{Rispetto a (1 0 2)}]{\text{Shuff. Vert}} \begin{pmatrix} 2 & 1 & 3 \\ 5 & 4 & 6 \\ 8 & 7 & 9 \end{pmatrix} \xrightarrow[\text{Rispetto a (0 2 1)}]{\text{Shuff. Vert}} \begin{pmatrix} 2 & 3 & 1 \\ 5 & 6 & 4 \\ 8 & 9 & 7 \end{pmatrix} \xrightarrow[\text{Rispetto a (1 0 2)}]{\text{Shuff. Oriz}} \begin{pmatrix} 5 & 6 & 4 \\ 2 & 3 & 1 \\ 8 & 9 & 7 \end{pmatrix} \xrightarrow[\text{Rispetto a (0 2 1)}]{\text{Shuff. Oriz}} \begin{pmatrix} 5 & 6 & 4 \\ 8 & 9 & 7 \\ 2 & 3 & 1 \end{pmatrix}$$

Scrivere un metodo Java--, chiamato **shuffleMatrice**, che dati in input una matrice quadrata T ed una matrice S definiti come precedentemente descritto, restituisca una nuova matrice data dall'applicazione di uno shuffle completo alla matrice T rispetto alla matrice S.

[**DIFFICOLTA' RIDOTTA**]: Si può assumere che la matrice S sia sempre composta da una sola riga (quindi si assuma che S sia sempre di dimensione 1 x *m*, con *m>0*). In questo caso lo shuffle completo consiste nell'applicazione alla matrice di partenza T di 1 singolo shuffle verticale e di 1 singolo shuffle orizzontale rispetto alla prima riga di S (si veda Esempio 1).

NOTA BENE:

- Gli studenti dovranno consegnare per questo esercizio solo 1 sorgente relativo alla soluzione con "difficoltà ridotta" oppure relativo alla soluzione con difficoltà standard.
- I Junit Test da superare per la difficoltà standard sono quelli della classe **ShuffleTest**.
- I Junit Test da superare nel caso di "difficoltà ridotta" sono solo quelli della classe **ShuffleRidottoTest** (quelli della classe ShuffleTest falliranno).
- 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.