– Simulazione di Prova Intermedia –

Corrisponde alla Prova Intermedia del 17/02/2023

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 del corso al seguente link:

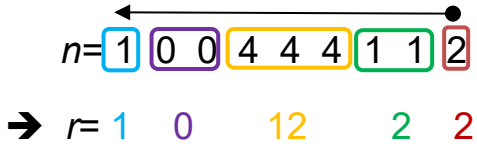
https://e-l.unifi.it/mod/assign/view.php?id=1262985

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: somma cifre ripetute

Sia *n* un numero intero (di tipo `int`) con *n* ≥ 0. La somma delle cifre ripetute nel numero *n* è una procedura che, partendo dalla cifra meno significativa di *n* e procedendo verso le cifre più significative, produce un nuovo numero intero in cui le sequenze di cifre ripetute vengono sostituite dalla somma delle cifre di cui sono composte.

Ad esempio, sia *n*=100444112, la somma delle cifre ripetute nel numero *n* genera il numero intero *r* =101222 poiché, leggendo le cifre da destra (cifre meno significative) verso sinistra (cifre più significative), si incontrano e si sommano le seguenti sequenze composte dalla stessa cifra ripetuta:



Altri esempi:

- Con *n*= 1111111111, la procedure genera l'intero *r*=10;
- Con *n*= 2 3 3 3 3 2 2, la procedure genera l'intero *r*=2124;
- Con *n*= 5 5 0 0 0 0, la procedure genera l'intero *r*=100;
- Con *n*= 9 3 3 3 2 2 2, la procedure genera l'intero *r*=996;
- Con *n*= 4 2 3 1 0, la procedure genera l'intero *r*=42310; *(es. 1)*
- Con *n*= 4 4 3 3 0, la procedure genera l'intero *r*=860; *(es. 2)*
- Con *n*= 1 1 2 2 3 4, la procedure genera l'intero *r*=2434. *(es. 3)*
-

[**DIFFICOLTA' STANDARD**]: Scrivere un metodo Java--, chiamato **sommaCifreRipetute**, che, dato in input un numero intero *n* (di tipo `int`), con *n*≥0, restituisca il numero intero (di tipo `int`) generato dalla somma delle cifre ripetute nel numero *n* come precedentemente descritto.

[**DIFFICOLTA' RIDOTTA**]: Si può assumere che le cifre di cui è composto il numero *n* siano tutte minori o uguali a 4, e che non possono esserci più di 2 cifre ripetute in ciascuna sequenza (vedere *es. 1, 2, 3*).

NOTA BENE:

- Saranno premiate le soluzioni che occuperanno meno memoria dati.
- 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.
- Gli studenti dovranno consegnare per questo esercizio solo 1 sorgente relativo alla soluzione con "difficoltà ridotta" oppure relativo alla soluzione con "difficoltà standard".

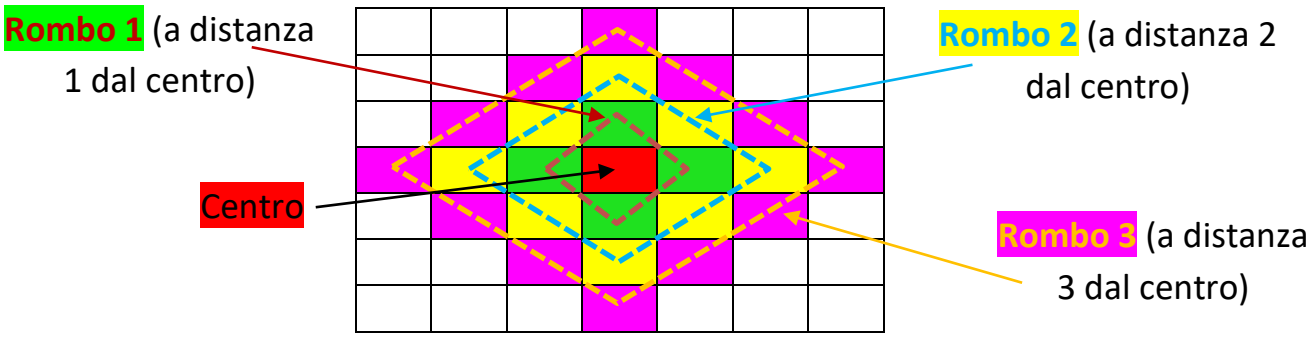
Junit Test:

- I Junit Test da superare per la "difficoltà ridotta" sono quelli della classe **SommaCifreRipetuteTest_Ridotta** (i test della classe SommaCifreRipetuteTest_Standard falliranno).
- I Junit Test da superare nel caso di "difficoltà standard" sono solo quelli della classe **SommaCifreRipetuteTest_Standard**.

Esercizio Java-- n. 2: rotazione rombo di matrice

Sia *T* una matrice quadrata di interi di dimensione *m* x *m*, con *m*≥*3* e *m* dispari. Sia *k* un intero, con 1≤ *k* ≤ *m*/2.

Informalmente, si definisce rombo *k*-mo della matrice *T* come l'insieme di celle di *T* che "distano" *k* celle dal centro della matrice e sono disposte a rombo come mostrato nella seguente figura:



Ad esempio, in una matrice *T* di dimensione 7x7 si possono definire tre rombi a distanza 1, 2 e 3 dal centro, come nella seguente figura (esempio 1):

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49

Una rotazione del rombo *k* della matrice *T* modifica la matrice ruotando in senso anti-orario di 1 posizione tutti gli elementi del rombo *k* della matrice. Ad esempio, la rotazione del rombo 2 della matrice dell'esempio precedente modifica la matrice in questo modo:

1	2	3	4	5	6	7
8	9	10	19	12	13	14
15	16	11	18	27	20	21
22	17	24	25	26	33	28
29	30	23	32	39	34	35
36	37	38	31	40	41	42
43	44	45	46	47	48	49

[**CONSEGNA OBBLIGATORIA**]: Scrivere un metodo Java--, chiamato **ruotaRomboMatrice**, che dati in input una matrice quadrata *T* di interi (di tipo `int`) di dimensione *m* x *m* (con *m*≥*3* e *m* dispari), ed un intero *k* (con 1≤*k*≤*m*/2), restituisca la matrice *T* modificata dalla rotazione del suo rombo *k* come precedentemente descritto.

[**CONSEGNA EXTRA - FACOLTATIVA**]: Scrivere un metodo Java--, chiamato **ruotaRomboMatriceNposizioni**, che dati in input una matrice quadrata *T* di interi di dimensione *m* x *m* con *m*≥*3* e *m* dispari, un intero *k* con 1≤*k*≤*m*/2, ed un numero intero *n*, restituisca la matrice *T* modificata dalla rotazione del suo rombo *k* di *n* posizioni in senso anti-orario (se n>0) o in senso orario (se n<0). Ad esempio, la rotazione del rombo *k*=2 della matrice *T* di partenza (vedi esempio 1) di 3 posizioni in senso anti-orario (quindi con *n*=3) modifica la matrice in questo modo:

1	2	3	4	5	6	7
8	9	10	33	12	13	14
15	16	27	18	39	20	21
22	19	24	25	26	31	28
29	30	11	32	23	34	35
36	37	38	17	40	41	42
43	44	45	46	47	48	49

NOTA BENE:

- I Junit Test da superare per la consegna obbligatoria sono quelli della classe **ruotaRomboMatriceTest** (quelli della classe ruotaAnelloMatriceNposizioniTest falliranno).
- I Junit Test da superare per la consegna extra (facoltativa) sono quelli della classe **ruotaRomboMatriceNposizioniTest**.
- 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.