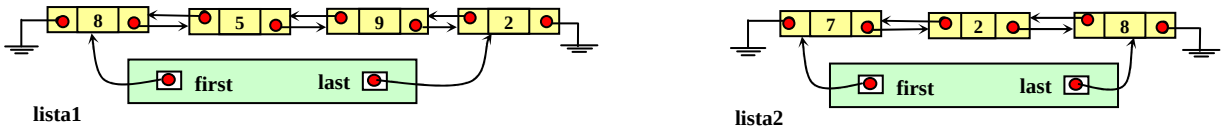
Esame scritto di:	Fondamenti di Informatica - corsi 4 e 5	**Data:**	12 dicembre 2001
Traccia:	A	**Tempo disponibile:**	3 ore

Cognome	Nome	Matricola	Corso	CORSO DI LAUREA

Esercizio 1.
Si consideri la seguente classe:

```
class ES1 {
        public static LinkedList process (LinkedList L1, LinkedList L2) {
                LinkedList lRis = new LinkedList();
                ListIterator it1 = L1.listIterator(),   it2 = L2.listIterator();
                for ( it1.end(); it1.hasCurrent(); it1.previous() ) {
                        boolean trovato = false;
                        for ( it2.begin();  it2.hasCurrent() && ! trovato;  it2.next() )
                                if ( L1.get(it1) == L2.get(it2) )
                                        trovato = true;
                        if ( !trovato )
                                lRis.addFirst(L1.get(it1));
                }
                return lRis;
        }
}
```

Si descriva il comportamento del metodo `process`, in generale; inoltre, con riferimento alle liste `lista1` e `lista2` riportate nella seguente figura



si commenti l'esecuzione dell'istruzione

```
LinkedList lista3 = ES1.process(lista1,lista2);
```

Esercizio 2.

```
class Couple {
        private int x;
        private int y;
        public Couple(int x1, int y1){
                x = x1;
                y = y1;
        }
        public void setZero() {
                x = 0;
                y = 0;
        }
        public String toString() {
                return "(" + x + "," + y + ")";
        }
}
class ES2 {
      public static void main ( String[] args ) {
            Couple a = new Couple (0,1);
            Couple b = new Couple (0,1);
            Couple c = a;
            if ( b == a )
                      System.out.println("b==a");
            else {
                      System.out.println("b!=a");
                      a.setZero();
            }
            if ( c == a )
                      System.out.println("c==a");
            else
                      System.out.println("c!=a");
            System.out.println("a: " + a);
            System.out.println("b: " + b);
            System.out.println("c: " + c);
      }
}
```

Si descriva il comportamento del metodo `main` della classe `ES2`: cosa viene stampato sul video?
Giustificare la risposta.

Esame scritto di:	Fondamenti di Informatica - corsi 4 e 5	**Data:**	12 dicembre 2001
Traccia:	**A**	**Tempo disponibile:**	3 ore

Esercizio 3.

Si definisca un metodo statico

```
public static double fAngolo ( double[] a, double[] b )
```

che, ricevuti due vettori a e b di uguale dimensione, restituisca la seguente misura:

$$\frac{a \otimes b}{\| a \| \cdot \| b \|}$$

dove $a \otimes b$ denota l'operazione di prodotto scalare fra vettori, cioè (indicata con n la dimensione dei vettori):

$$a \otimes b = \sum_{i=1}^{n} a_i \cdot b_i$$

mentre, dato un generico vettore x, $\| x \|$ denota la norma di x, calcolata come segue:

$$\|x\| = \sqrt{\sum_{i=1}^{n} x_i^{\,2}}$$

Esercizio 4.

Si definisca un metodo statico

```
public static boolean confrontaSequenze ( int[][] a )
```

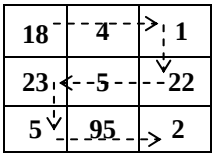
che, data una matrice a quadrata di interi, restituisca *true* se sono uguali le due sequenze *sequenzaOrizzontale* e *sequenzaVerticale* ottenute leggendo gli elementi della matrice a "zig-zag" in orizzontale e in verticale, rispettivamente, come indicato nell'esempio successivo; se le due sequenze sono differenti il metodo deve restituire *false*.

Ad esempio, se *a* è la matrice

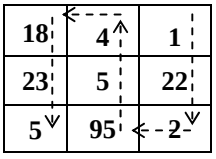
18	**4**	**1**
23	**5**	**22**
5	**95**	**2**

le due sequenze sono così definite:

- *sequenzaOrizzontale* è ottenuta visitando gli elementi della matrice come indicato nella seguente figura



- *sequenzaVerticale* è ottenuta visitando gli elementi della matrice come indicato nella seguente figura



Pertanto, in tale esempio, la sequenza *sequenzaOrizzontale* è [18, 4, 1, 22, 5, 23, 5, 95, 2] mentre la sequenza *sequenzaVerticale* è [1, 22, 2, 95, 5, 4, 18, 23, 5].

TRACCIA A - SOLUZIONE

Esercizio 1.
Il metodo *process* della classe *ES1* riceve due liste *L1* e *L2* (oggetti della classe *LinkedList*) e restituisce una lista contenente gli elementi della lista *L1* che non sono presenti nella lista *L2* (se nessuna delle due liste contiene elementi duplicati, indicando con *S1* [rispettivamente *S2*] l'insieme degli elementi della lista *L1* [rispettivamente *L2*], allora la lista risultato contiene la differenza insiemistica *S1*-*S2*).
Gli elementi della lista risultato sono disposti in sequenza nello stesso ordine con cui appaiono nella lista di origine, poiché benché il metodo scandisca gli elementi di *L1* al contrario (dall'ultimo al primo), esso effettua inserimenti in testa.

In particolare, tramite il *for* esterno si scandiscono gli elementi della lista *L1* dall'ultimo al primo.
Per ogni elemento *x* di *L1*, tramite il *for* interno, si effettua una scansione della lista *L2* verificando se è presente almeno un elemento di *L2* uguale a *x*; *x* viene inserito in testa alla lista risultato (riferita da *lRis*) solo se la verifica ha esito negativo (*found==false*), cioè nessun elemento di *L1* è uguale a *x*.

Nell'esempio indicato nella traccia (dove le liste riferite da *lista1* e *lista2* contengono, rispettivamente, le sequenze <8,5,9,2> e <7,2,8>), dopo l'istruzione

```
LinkedList lista3 ES1_A.process(lista1,lista2);
```

lista3 riferisce una lista contenente la sequenza <5,9>

Esercizio 2.
L'output del programma è il seguente:

```
b != a
c == a
a: (0,0)
b: (0,1)
c: (0,0)
```

Infatti:

```
class ES2 {
	public static void main ( String[] args ) {
		// i riferimenti a e b puntano a oggetti distinti, creati mediante diverse invocazioni
		// dell'operatore new e inizializzati (con uguale stato) dal costruttore della classe Couple
		Couple a = new Couple (0,1);
		Couple b = new Couple (0,1);
		// il riferimento c punta allo stesso oggetto puntato dal riferimento a
		Couple c = a;
		// confronto fra riferimenti: l'esito è false, perché i riferimenti a e b puntano a
		// due oggetti distinti (anche se tali oggetti hanno lo stesso stato)
		// viene stampato "b != a" e viene cambiato lo stato dell'oggetto riferito da a (...e da c !!!)
		if ( b == a )
			System.out.println("b == a");
		else {
			System.out.println("b != a");
			a.setZero(); // pone a zero le componenti x e y dell'oggetto riferito da a (... e da c)
		}
		// confronto fra riferimenti: l'esito è true, perché a e c puntano allo stesso oggetto;
		// viene stampato "c == a"
		if ( c == a )
			System.out.println("c == a");
		else
			System.out.println("c != a");
		System.out.println("a: " + a); // stampa "a: (0,0)"
		System.out.println("b: " + b); // stampa "b: (0,1)"
		System.out.println("c: " + c); // stampa "c: (0,0)" (c riferisce lo stesso oggetto puntato da a)
    }
}
```

Esercizio 3.
Osservando, dato un vettore $\boldsymbol{x}$, il prodotto scalare $\boldsymbol{x} \otimes \boldsymbol{x}$ rappresenta il quadrato della norma di $\boldsymbol{x}$, una possibile soluzione dell'esercizio è la seguente:

```
public class ES3_A {

    static double prodottoScalare(double[] a, double[] b) {
       double ret=0;
       for(int i=0; i<a.length; i++)
            ret+=a[i]*b[i];
       return ret;
    }
```

```
    public static double fAngolo( double[] a, double[] b) {
        double normaA = Math.sqrt(prodottoScalare(a,a));
        double normaB = Math.sqrt(prodottoScalare(b,b));
        return prodottoScalare(a,b) / (normaA*normaB);
    }
}
```

Esercizio 4.
Dell'esercizio si forniscono due soluzioni alternative, la prima delle quali è la più compatta ed elegante.

SOLUZIONE 4.1
L'algoritmo proposto si basa sull'osservazione che esiste una relazione fra gli indici (di riga e di colonna) degli elementi che occupano la stessa posizione nelle sequenze; infatti, indicata con *n* la dimensione della matrice, l'elemento *a(i,j)* occupa nella sequenza orizzontale la stessa posizione occupata nella sequenza verticale dall'elemento $a(i_1,j_1)$, dove:

$i_1 = j$
$j_1 = n\text{-}1\text{-}i$

Per risolvere il problema è sufficiente verificare che nelle due sequenze non esista una coppia di elementi corrispondenti che abbiano valore diverso.

```
public static boolean sequenzeUguali ( int[][] a ){
        boolean finoraUguali = true;
        for ( int i=0; i < a.length && finoraUguali; i++ )
                for ( int j=0; j < a.length && finoraUguali; j++ )
                        if ( a[i][j] != a[j][a.length-1-i] )
                                finoraUguali = false;
        return finoraUguali;
}
```

SOLUZIONE 4.2
Una soluzione alternativa, meno efficiente, consiste nel memorizzare le due sequenze due array e, quindi, confrontare il contenuto degli array.

```
class ES2 {
        public static boolean sequenzeUguali ( int[][] a ) {
                int[] seqOriz = generaSeqOriz(a);    // vettore per la sequenza orizzontale
                int[] seqVert = generaSeqVert(a);    // vettore per la sequenza verticale
                boolean uguali = true;
                for( int i = 0; i < seqOriz.length && uguali; i++)
                        if ( seqOriz[i] != seqVert[i] )
                                uguali = false;
                return uguali;
        }
        private static int[] generaSeqOriz ( int[][] m ) {
                int[] v = new int [m.length*m.length];
                int pos = 0;          // indice per scrivere gli elementi del vettore
                int riga, col, k;     // indici per leggere gli elementi della matrice
                for( riga = 0; riga < m.length ; riga ++ )
                        for ( int k=0; k < m.length; k++) {
                                if ( riga % 2 == 0 )
                                        //scansione per indice di colonna crescente per le righe pari
                                        col = k;
                                else
                                       //scansione con indice di colonna decrescente per le righe dispari
                                        col = m.length -1-k;
                                v[pos] = m[riga][col];
                                pos++;
                        }
                return v;
        }
        private static int[] generaSeqVert ( int[][] m ) {
                int[] v = new int [m.length*m.length];
                int riga, col, k;     // indici per leggere gli elementi della matrice
                int pos = 0;          // indice per scrivere gli elementi del vettore
                for( col = m.length –1 ; col >= 0; col-- )
                        for ( int k=0; k < m.length; k++) {
                                if ( (m.length–1-col) % 2 == 0 )
                                        riga = k;
                                else
                                        riga = m.length-1-k;
                                v[pos] = m[riga][col];
                                pos++;
```

```
                }
        return v;
    }

    // metodo main per testare la classe
    public static void main ( String[] args ) {
                int [][] a = {{1,2,2,1},{2,4,4,2},{2,4,4,2},{1,2,2,1}};
                if ( ES4.sequenzeUguali(a) )
                        System.out.println("Le due sequenze sono uguali.");
                else
                        System.out.println("Le due sequenze non sono uguali.");
    }
}
```

SOLUZIONE 4.3

```
    public static boolean sequenzeUguali (int[][] a){
                int i = 0, j = 0;            //indici per la scansione orizzontale
                int k = 0, p = a.length-1;   //indici per la scansione verticale
                /* variabili  booleane per indicare la direzione di avanzamento sulle righe della sequenza
                   orizzontale e sulle colonne della sequenza verticale */
                boolean avantiOriz = true;
                boolean avantiVert = true;
                boolean finoraUguali = true;
                while ( i<a.length && finoraUguali ) {
                        // verifica uguaglianza elementi corrispondenti sulle due sequenze
                        if ( a[i][j] != a[k][p] )
                                finoraUguali = false;
                        else {
                                //avanzamento sequenza orizzontale
                                if ( avantiOriz ) {
                                        if ( j==a.length-1 ){ //fine riga
                                                i++;
                                                avantiOriz = false;
                                        }
                                        else
                                                j++;
                                }
                                else {
                                        if ( j==0 ){   //fine riga
                                                i++;
                                                avantiOriz = true;
                                        }
                                        else
                                                j--;
                                }
                                //avanzamento sequenza verticale
                                if ( avantiVert ) {
                                        if ( k==a.length-1 ) {
                                                p--;
                                                avantiVert = false;
                                        }
                                        else
                                                k++;
                                }
                                else {
                                        if ( k==0 ) {
                                                p--;
                                                avantiVert = true;
                                        }
                                        else
                                                k--;
                                }
                        } // end if
                } //end while
                return finoraUguali;
    }
```