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

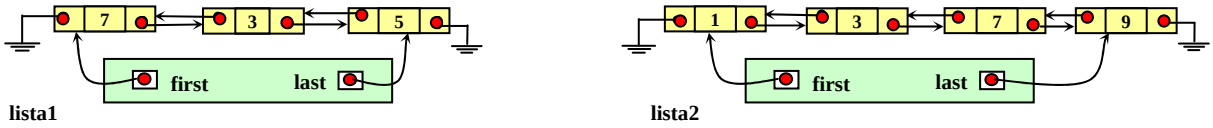
Cognome	Nome	Matricola	Corso	CORSO DI LAUREA

Esercizio 1.

Si consideri la seguente classe:

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

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



si commenti l'esecuzione dell'istruzione

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

Esercizio 2.

```
class Pair {
        private int a;
        private int b;
        public Pair () {
                a = 0;
                b = 0;
        }
        public void set(int a1, int b1) {
                a = a1;
                b = b1;
        }
        public String toString() {
                return "[" + a + " - " + b + "]";
        }
}
class ES2 {
      public static void main ( String[] args ) {
            Pair y = new Pair();
            Pair w = new Pair();
            Pair z = w;
            if ( y == w )
                      System.out.println("y == w");
            else {
                      System.out.println("y != w");
                      w.set(4,5);
            }
            if ( z == w )
                      System.out.println("z == w");
            else
                      System.out.println("z != w");
            System.out.println("w = " + w);
            System.out.println("y = " + y);
            System.out.println("z = " + z);
      }
}
```

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:	**B**	**Tempo disponibile:**	3 ore

Esercizio 3.

Si definisca un metodo statico

```
public static double misura ( double[] x, double[] y, double[] z )
```

che, ricevuti tre vettori x, y e z di uguale dimensione, restituisca la seguente misura:

$$\frac{x \otimes y}{\| z \|}$$

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

$$x \otimes y = \sum_{i=1}^{n} x_i \cdot y_i$$

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

$$\|a\| = \sqrt{\sum_{i=1}^{n} a_i^{\,2}}$$

Esercizio 4.

Si definisca un metodo statico

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

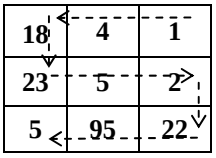
che, ricevuta una matrice a quadrata di interi, restituisca *true* se sono uguali le due sequenze *sequenzaOrizzontale* e *sequenzaVerticale* ottenute leggendo gli elementi della matrice a "zig-zag" in orizzontale e in verticale, rispettivamente, come indicato nell'esempio successivo; se le due sequenze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

Ad esempio, se a è la matrice

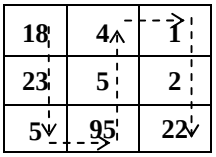
18	**4**	**1**
23	**5**	**22**
5	**95**	**2**

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 [1, 4, 18, 23, 5, 2, 22, 95, 5] mentre la sequenza *sequenzaVerticale* sarà [18, 23, 5, 95, 5, 4, 1, 22].

TRACCIA B - SOLUZIONE

Esercizio 1.
Il metodo *elaborate* della classe *ES1* riceve due liste *L1* e *L2* (oggetti della classe *LinkedList*) e restituisce una lista contenente gli elementi della lista L2 che non sono presenti nella lista L1 (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 *S2-S1*).
Gli elementi della lista risultato sono disposti in sequenza in ordine inverso rispetto a come appaiono nella lista di origine *L2*, poiché il metodo scandisce gli elementi di *L2* dal primo all'ultimo ma li inserisce in testa nella lista risultato.

In particolare, tramite il *for* esterno si scandiscono gli elementi della lista *L2* dal primo all'ultimo.
Per ogni elemento *x* di *L2*, tramite il *for* interno, si effettua una scansione della lista *L1* (al contrario, ma ciò è irrilevante ai fini del risultato) verificando se è presente almeno un elemento di *L1* con lo stesso valore di *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 (le liste *lista1* e *lista2* contengono, rispettivamente, le sequenze <7,3,5> e <1,3,7,9>), dopo l'istruzione

```
LinkedList lista3 ES1_B.elaborate(lista1,lista2);
```

lista3 riferisce una lista contenente la sequenza <9,1>.

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

y != w
z == w
w = [4 - 5]
y = [0 - 0]
z = [4 - 5]

Infatti :

```
class Pair {
	private int a;
	private int b;
	public Pair () {
		a = 0;
		b = 0;
	}
	public void set(int a1, int b1) {
		a = a1;
		b = b1;
	}
	public String toString() {
		return "[" + a + " - " + b + "]";
	}
}

class ES2 {
	public static void main ( String[] args ) {
		// i riferimenti w e y puntano a oggetti distinti, creati mediante diverse invocazioni
		// dell'operatore new e inizializzati (con uguale stato) dal costruttore della classe Pair
		Pair y = new Pair();
		Pair w = new Pair();
		// il riferimento z punta allo stesso oggetto puntato dal riferimento w
		Pair z = w;
		//  confronto fra riferimenti: l'esito è false, perché i riferimenti w e y puntano a  due oggetti distinti
		// (anche se tali oggetti hanno lo stesso stato);
		//  viene stampato "y != w" e viene cambiato lo stato dell'oggetto riferito da w (...e da z !!!)
		if ( y == w )
			System.out.println("y == w");
		else {
			System.out.println("y != w");
			w.set(4,5); //pone rispettivamente a 4 e 5 le componenti a e b dell'oggetto riferito da w (... e da z)
		}
		//  confronto fra riferimenti: l'esito è true, perché z e w puntano allo stesso oggetto;  viene stampato "z == w"
		if ( z == w )
			System.out.println("z == w");
		else
			System.out.println("z != w");
		System.out.println("w = " + w); // stampa " w = [4 - 5] "
		System.out.println("y = " + y); // stampa " y = [0 - 0] "
		System.out.println("z = " + z); // stampa " z = [4 - 5] " (z riferisce lo stesso oggetto puntato da w)

	}
}
```

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 {

    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 misura( double[] a, double[] b, double[] c ) {
        double normaC = Math.sqrt(prodottoScalare(c,c));
        return prodottoScalare(a,b) / normaC;
    }

}
```

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, elemento $a(i,j)$ occupa nella sequenza verticale la stessa posizione occupata nella sequenza orizzontale 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 confrontaSequenze ( 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 confrontaSequenze ( 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 = m.length-1; k > 0; k--) {
                              if ( riga % 2 == 0 )
                                      //scansione per indice di colonna decrescente per le righe pari
                                      col = k;
                              else
                                      //scansione per indice di colonna crescente 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 = 0; col < m.length; col++ )
                        for ( int k=0; k < m.length; k++ ) {
                                if ( col % 2 == 0 )
                                        //scansione per indice di riga crescente per le colonne pari
                                        riga = k;
                                else
                                        //scansione per indice di riga decrescente per le colonne dispari
                                        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 confrontaSequenze ( int[][] a ){
                //indici per la scansione orizzontale
                int i = 0, j = a.length-1;
                //indici per la scansione verticale
                int k = 0, p = 0;
                /*  booleani per indicare la direzione di avanzamento sulle righe della sequenza
                    orizzontale e sulle colonne della sequenza verticale */
                boolean avantiOriz = false;
                boolean avantiVert = true;

                boolean finoraUguali = true;
                while ( i<a.length && finoraUguali ) {
                        if ( a[i][j] != a[k][p] )
                                finoraUguali = false;
                        else {
                                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--;
                                }
                                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;
        }
}
```