Prova scritta di Programmazione II

FAC-SIMILE

LEGGERE CON ATTENZIONE

- Il tempo a disposizione per lo svolgimento della prova è di **2 ore**.
- Non è consentita la consultazione di appunti, dispense, libri, ecc.
- Non è consentito l'uso di dispositivi quali laptop, tablet, smartphone, e-reader, ecc.
- Al termine della prova, **consegnare** il testo del compito e tutti i fogli protocollo contenenti esercizi da correggere.
- Ricordarsi di **scrivere nome, cognome e matricola** su ogni foglio protocollo consegnato.

DATI DELLO STUDENTE

Nome	
Cognome	
Matricola	
Corso (A o B)	

SEZIONE RISERVATA AL DOCENTE

Esercizio 1	
Esercizio 2	
Esercizio 3	
Esercizio 4	

Esercizio 1 *Data la classe*

```
class Node<T> {
    public T elem;
    public Node<T> next;

    public Node(T elem, Node<T> next) {
        this.elem = elem;
        this.next = next;
    }
}
```

implementare un metodo statico ricorsivo

```
    public static <T> boolean controlla(Node<T> p, Node<T> q) {
      ...
    }
```

che restituisce `true` *se esiste un elemento* x *che occorre in entrambe le liste* `p` *e* `q` *nella stessa posizione. Ad esempio, avremo:*

- `controlla([5, 7], [8, 7, 1]) == true`
- `controlla([true, false, false], [false, true, false]) == true`
- `controlla([3, null, 4], [1, null]) == true`
- `controlla([1, 2], [2, 1]) == false`
- `controlla([], ["ciao"]) == false`

Esercizio 2 *Date le classi*

```
abstract class A {
    public abstract void m1();
}

abstract class B extends A {
    public void m1()
    { System.out.println("B.m1"); }

    public abstract void m2(A obj);
}

class C extends B {
    public void m1()
    {
        System.out.println("C.m1");
        super.m1();
    }

    public void m2(A obj)
    {
        System.out.println("B.m2");
        obj.m1();
    }
}
```

rispondere alle seguenti domande:

1. *Se si eliminasse il metodo* `m2` *dalla classe B, il codice sarebbe comunque corretto? Perché?*
2. *Il seguente codice è corretto? Se no, spiegare perché. Se sì, determinare cosa stampa.*

   ```
   A obj = new B();
   obj.m2(obj);
   ```

3. *Il seguente codice è corretto? Se no, spiegare perché. Se sì, determinare cosa stampa.*

   ```
   A obj = new C();
   obj.m1();
   ```

4. *Il seguente codice è corretto? Se no, spiegare perché. Se sì, determinare cosa stampa.*

   ```
   A obj = new C();
   obj.m2(obj);
   ```

Esercizio 3 (6 punti) *Sia dato il metodo*

```
public static <T> boolean m4(T[][] a) {
    for (int i = 0; i < a.length; i++)
        for (int j = 0; j < a.length; j++)
            if (!a[i][j].equals(a[j][i])) return false;
    return true;
}
```

1. *Descrivere in modo conciso e chiaro, in* ***non più di 2 righe di testo****, l'effetto del metodo.*
2. *Determinare la condizione* ***più debole*** *che garantisce l'esecuzione del metodo senza eccezioni e scrivere una corrispondente* ***asserzione*** *da aggiungere come precondizione per il metodo. Nello scrivere l'asserzione è possibile fare uso di eventuali metodi statici ausiliari che* ***vanno comunque definiti*** *anche se visti a lezione.*

1) il metodo scorre una matrice di oggetti e controlla se ogni elemento della matrice è uguale all'opposto rispetto alla diagonale della matrice. Il metodo termina quando trova un elemento diverso dal suo opposto oppure quando tutti gli elementi sono uguali al suo opposto

2) la condizione è che a == null e a deve essere quadrata perciò come asserzione metterei: assert a == null && isQuadrata(a) : "Non è possibile verificare la matrice"

```
public static <T> boolean isQuadrata(T[][] a){
    for(int i = 0; i < a.length; i++)
            if(!a[i].length == a[0].length) return false;
   return true;
}
```

Esercizio 4 *Date le classi*

```
abstract class Tree {
    public abstract Tree insert(int elem);
}

class Leaf extends Tree {
    public Tree insert(int elem)
    { return new Branch(elem, this, this); }
}

class Branch extends Tree {
    private int elem;
    private Tree left;
    private Tree right;

    public Branch(int elem, Tree left, Tree right) {
        this.elem = elem;
        this.left = left;
        this.right = right;
    }

    public Tree insert(int elem) {
        /***** CHECK POINT 1 *****/
        if (elem < this.elem) left = left.insert(elem);
        else if (elem > this.elem) right = right.insert(elem);
        return this;
    }
}

class TestHeap {
    public static void main(String[] args) {
        Tree t = new Leaf();
        t = t.insert(3);
        t = t.insert(1);
        t = t.insert(4);
        t = t.insert(2);
        /***** CHECK POINT 2 *****/
    }
}
```

si disegni una rappresentazione dello stato della memoria (STACK e HEAP)

1. *la prima volta che l'esecuzione raggiunge il check point 1;*
2. *al check point 2.*