| ITIS Belluzzi<br>a.s. 2013/14 | **Verifica di Informatica**<br>**21/1/14** | Classe:<br>4Ai | Nome e cognome: |
|---|---|---|---|

# Implementazione di Lista Ordinata

Si consideri la seguente interfaccia:

```
/**
 * Una lista concatenata che ha i dati in ordine crescente
 * Nota Bene: <T extends Comparable<T>> implica che T dispone del metodo compareTo()
 */
public interface ListaOrdinata<T extends Comparable<T>> {
  /**
   * Inserisce il dato nella posizione che garantisce l'ordinamento crescente
   * @param dato il dato da inserire
   */
  public void inserisci(T dato);
  /**
   * Restituisce la posizione del dato (a partire da 0 che indica il primo elemento)
   * @param dato il dato da cercare
   * @return la posizione trovata, si il dato manca restituisce -1
   */
  public int posto(T dato);
  /**
   * Rimmuove il dato  dalla lista, se il dato manca fa nulla
   * @param dato il dato da rimuovere
   */
  public void rimuovi(T dato);
  /**
   * Elimina ogni dato dalla lista
   */
  public void svuota();
  /**
   * @return il numero di dati presenti in lista
   */
  public int lunghezza();
}
```

Si intende realizzare la classe Elenco come implementazione di ListaOrdinata:

```
/**
 * Una implementazione di Listaordinata
 */
public class Elenco<T extends Comparable<T>> implements ListaOrdinata<T> {
  Nodo<T> testa;
  public Elenco(){...}
  //implementazione dei diversi metodi...
  private class Nodo<T>{...}
}
```

Al fine di implementare completamente la ListaOrdinata si richiede:

- (*2 punti*) sviluppo del costruttore di Elenco e della inner class Nodo
- (*1 punto ciascuno*) implementazione dei metodi di interfaccia
- (2 punti) la classe Elenco implementa anche Iterable<T>

# Implementazione di Stack

(*3 punti*) Si implementi l'interfaccia Stack<T> facendo uso della classe ListaConcatenata<T>
Promemoria:

```
public interface Stack<T> {
  void push(T dato);
  void pop();
  T top();
  boolean isEmpty();
}
```