# Esame di Fondamenti di Informatica L-B
# Ingegneria Gestionale e dei Processi Gestionali (L-Z)

## Appello del 13/9/2012

***Esercizio 1 (4 punti)***

Illustrare il concetto di complessità di un algoritmo.

***Esercizio 2 (6 punti)***

Siano dati i seguenti metodi Java:

```
public static int f(int V[], int N) {
  int sum=0, i=0;
  while(++i<N)
    sum+=V[i++];
  return sum;
}

public static int g(int V[],int N) {
  int j=0, sum=0;
  do
    sum+=f(V, ++j);
  while(j++<N)
  return sum;
}
```

1. Calcolare la complessità in passi base del metodo `f` nei termini del parametro `N` (si distinguano i casi in cui `N` assume valori pari da quelli in cui assume valori dispari).
2. Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N` pari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

***Esercizio 3 (6 punti)***

L'agenzia di viaggi "A.B. Road" organizza vacanze per singoli e gruppi. Un viaggio si compone tipicamente di una serie di tappe, ciascuna caratterizzata dal luogo e dalla durata della permanenza. Per comodità degli impiegati dell'agenzia, nel sistema vengono memorizzate anche alcune note testuali. Si scriva una classe `Tappa` per il sistema informativo della "A.B. Road" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca una descrizione della tappa.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Tappa` (l'uguaglianza va verificata sul luogo e sulla durata della permanenza).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Tappa` passato come parametro (la precedenza va data per ordine decrescente di durata, in caso di parità si utilizzi l'ordine alfabetico sul luogo).

***Esercizio 4 (7 punti)***

Si scriva una classe `Viaggio` che memorizzi le informazioni relative ad un viaggio organizzato dalla "A.B. Road". Per ogni viaggio si memorizzi una descrizione sintetica (tipo "viaggio di nozze"), il numero di persone che partecipano ed una sequenza di tappe (sono possibili duplicati). La classe `Viaggio` deve inoltre:

1. Presentare un opportuno costruttore con parametri (inizialmente, un viaggio non ha tappe).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del viaggio (inclusa la descrizione di tutte le tappe ad esso associate).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Tappa`, lo inserisca in coda alla lista.
5. Presentare il metodo `durata` che restituisca la durata totale del viaggio.
6. Possedere il metodo `principale` che restituisca la tappa in cui i viaggiatori si soffermeranno il maggior numero di giorni (in caso di parità, si utilizzi l'ordine alfabetico sul luogo).

***Esercizio 5 (8 punti)***

Si scriva un'applicazione per la ditta "A.B. Road" che:

1. Crei un insieme di oggetti `Viaggio`.
2. Crei un oggetto `Viaggio`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. all'interno dell'insieme di cui al punto 1., controllando che tale inserimento sia possibile.
4. Crei un oggetto `Tappa`, lette da tastiera le informazioni necessarie.
5. Inserisca la tappa creata al punto 4. tra quelle associate al viaggio di cui al punto 2.
6. Stampi a video la descrizione del viaggio di maggior durata tra quelli dell'insieme di cui al punto 1 (a parità di durata si selezioni il viaggio con più partecipanti).
7. Stampi a video la tappa principale del viaggio di cui al punto 6.

***Soluzione Esercizio 2***

**Domanda 1:**

| 2 assegnamenti | 2 | o 2 |
|---|---|---|
| `++i<N` | N/2 + 1 | o (N+1)/2 |
| `sum+=V[i++]` | N/2 | o (N–1)/2 |
| Totale | N + 3 | o N + 2 |

**Domanda 2:**

| 2 assegnamenti | 2 |
|---|---|
| `sum+=f(V, ++j)` | N/2 + 1 |
| `while(j++<N)` | N/2 + 1 |
| complessità di `f` | $N^2/4 + 2N + 3$ |
| Totale | $N^2/4 + 3N + 7$ |

complessità di `f`: $\sum_{\substack{j=1 \\ (j \text{ dispari})}}^{N+1} (j+2) = \sum_{i=0}^{N/2}(2i+3) = \frac{N^2}{4} + 2N + 3$

**Domanda 3:**
Complessità asintotica: $O(N^2)$

***Soluzione Esercizio 3***

```
class Tappa implements Comparable<Tappa> {
  private String luogo, note;
  private int durata;

  public Tappa (String luogo, int durata, String note) {
    this.luogo=luogo;
    this.durata=durata;
    this.note=note;
  }

  public String getLuogo() { return luogo; }
  public int getDurata() { return durata; }
  public String getNote() { return note; }

  public String toString() {
    return luogo + ", per " + durata + " giorni (" + note + ")";
  }

  public boolean equals(Object o) { return equals((Tappa) o); }
  public boolean equals(Tappa t) {
    return luogo.equals(t.luogo) && durata==t.durata;
  }

  public int compareTo(Tappa t) {
    int ret=t.durata-durata;
    if(ret==0) ret=luogo.compareTo(t.luogo);
    return ret;
  }
}
```

***Soluzione Esercizio 4***

```
import java.util.*;

class Viaggio {
  private String descrizione;
  private int numeroPersone;
  private List<Tappa> l;

  public Viaggio(String descrizione, int numeroPersone) {
    this.descrizione=descrizione;
    this.numeroPersone=numeroPersone;
    l=new ArrayList<Tappa>();
  }

  public String getDescrizione() { return descrizione; }
  public int getNumeroPersone() { return numeroPersone; }
  public String toString() {
    return descrizione + ", per " + numeroPersone + ": " + l.toString();
  }

  public void aggiungi(Tappa t) {
    l.add(t);
  }

  public int durata() {
    int durata=0;
    for (Tappa t: l) durata+=t.getDurata();
    return durata;
  }

  public Tappa principale() {
    Tappa max=null;
    for (Tappa t: l)
      if (max==null || t.compareTo(max)<0) max=t;
    return max;
  }
}
```

***Soluzione Esercizio 5***

```
import java.util.*;
import fiji.io.*;
class Applicazione {
  public static void main(String[] args) {
    Set<Viaggio> s=new HashSet<Viaggio>();
    Viaggio v=new Viaggio(Lettore.in.leggiLinea(), Lettore.in.leggiInt());
    if(!s.add(v)) System.out.println("Viaggio già presente!");
    Tappa t=new Tappa(Lettore.in.leggi(), Lettore.in.leggiInt(),
      Lettore.in.leggiLinea());
    v.aggiungi(t);
    Viaggio max=null;
    int maxD=-1;
    for(Viaggio x: s) {
      int durata=x.durata();
      if(durata>maxD || (durata==maxD &&
          x.getNumeroPersone()>max.getNumeroPersone())) {
        max=x; maxD=durata;
      }
    }
    System.out.println(max.getDescrizione());
    System.out.println(max.principale());
  }
}
```