Esame di Fondamenti di Informatica T-1/T-A
Ingegneria Gestionale (A-K)

Appello del 17/2/2014

NOTA: Per il superamento dell'esame è **necessario** ottenere la sufficienza nello svolgimento dell'Esercizio 1.

Esercizio 1 (4 punti)

1. Discutere i diversi tipi di complessità computazionale di un algoritmo.
2. Discutere le differenze tra un linguaggio interpretato ed uno compilato.

Esercizio 2 (2 punti)

Convertire in binario i numeri **125** e **–86**, supponendo di utilizzare una rappresentazione a 8 bit in complemento a 2. Si esegua infine la somma dei due numeri, riconvertendo il risultato in base 10, motivando eventuali differenze tra il risultato ottenuto e quello atteso.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

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

public static int g(int V[], int N) {
  int j, sum=0;
  for (j=1; j<N; ++j)
    sum+=f(V, N, j);
  return sum;
}
```

1. Calcolare la complessità in passi base del metodo `f` nei termini dei parametri `M` e `N` (si distinguano i casi in cui `M` assume valori maggiori di `N` da quelli in cui assume valori minori o uguali a `N`).
2. Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N` dispari e maggiore di 0).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (5 punti)

La televisione di stato della repubblica caraibica di St. Marquez si sta preparando per il festival nazionale della canzone. In tale competizione uno stesso brano musicale viene interpretato da più cantanti. Ogni canzone viene pertanto memorizzata all'interno di un calcolatore, registrandone il titolo, la durata (in minuti) e il cognome dell'autore. Si scriva una classe `Canzone` 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 canzone.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Canzone` (la verifica va fatta sul titolo e sul cognome dell'autore).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Canzone` passato come parametro (in ordine alfabetico per titolo e, a parità, per durata crescente).

Esercizio 5 (7 punti)

Si scriva una classe `Cantante` che memorizzi le informazioni relative a un cantante partecipante al festival. Per ogni cantante occorre memorizzare il nome, mentre i brani eseguiti vanno inseriti all'interno di una lista. La classe `Cantante` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la lista dei brani eseguiti è vuota).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del cantante (inclusa la descrizione di tutti i brani eseguiti).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Canzone`, lo inserisca all'interno della lista, mantenendo la lista ordinata secondo il punto 5. dell'esercizio 4.
5. Presentare il metodo `esegue` che, data una canzone, indichi se tale cantante eseguirà o meno tale canzone al festival.
6. Possedere il metodo `autore` che, dato il cognome di un autore, restituisca il numero di brani scritti da tale autore all'interno del repertorio del cantante.
7. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Cantante` (la verifica va fatta esclusivamente sul nome).

Esercizio 6 (8 punti)

Si scriva un'applicazione per il festival della canzone di St. Marquez che:

1. Crei un insieme di oggetti `Cantante`.
2. Crei un oggetto `Cantante`, 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 `Canzone`, lette da tastiera le informazioni necessarie.
5. Inserisca la canzone creata al punto 4. tra quelle eseguite dal cantante di cui al punto 2.
6. Stampi a video il nome di tutti i cantanti, tra quelli dell'insieme di cui al punto 1., che eseguiranno il brano di cui al punto 4.
7. Letto da tastiera il cognome di un autore, stampi a video il nome del cantante che esegue più brani scritti da tale autore.
8. Relativamente all'autore di cui al punto 7., stampi a video il numero totale di brani di tale autore che verranno eseguiti.

Soluzione Esercizio 2

$125_{10} = 01111101_2$
$-86_{10} = 10101010_2$

$00100111_2 = 39_{10}$

```
01111101+
10101010
00100111
```

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`i-->N`	`M` – `N` + 1	o 1
`sum+=V[i]`	`M` – `N`	o 0
Totale	2`M` – 2`N` + 3	o 3

Domanda 2:

2 assegnamenti	2
`j<N`	`N`
`sum+=f(V, N, j)`	`N` – 1
`j++`	`N` – 1
complessità di `f`	N^2 + 2`N` – 3
Totale	N^2 + 5`N` – 3

complessità di `f`: $\sum_{j=1}^{N-1}(2N-2j+3) = 2\,N^2 - 2N - N^2 + N + 3N - 3 = N^2 + 2N - 3$

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

Soluzione Esercizio 4

```
class Canzone implements Comparable<Canzone> {
  private String titolo, autore;
  private float durata;

  public Canzone(String titolo, String autore, float durata) {
    this.titolo = titolo;
    this.autore = autore;
    this.durata = durata;
  }

  public String getTitolo() { return titolo; }
  public String getAutore() { return autore; }
  public float getDurata() { return durata; }

  public String toString() {
    return titolo + " (" + autore + "): " + durata;
  }

  public boolean equals(Object o) { return equals((Canzone) o); }
  public boolean equals(Canzone c) {
    return titolo.equals(c.titolo) && autore.equals(c.autore);
  }

  public int compareTo(Canzone c) {
    int ret = titolo.compareTo(c.titolo);
    if(ret==0) if(durata < c.durata) ret = -1;
      else if(durata > c.durata) ret = 1;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Cantante {
  private String nome;
  private List<Canzone> l;

  public Cantante(String nome) {
    this.nome = nome;
    l=new ArrayList<Canzone>();
  }

  public String getNome() { return nome; }
  public String toString() {
    return nome + "):" + l.toString();
  }

  public void aggiungi(Canzone c) {
    int i=0;
    while((i<l.size())&&(l.get(i).compareTo(c)<0)) i++;
    l.add(i, c);
  }

  public boolean esegue(Canzone c) { return l.contains(c); }

  public int autore(String autore) {
    int n = 0;
    for(Canzone c: l) if(c.getAutore().equals(autore)) n++;
    return n;
  }

  public boolean equals(Object o) { return equals((Cantante) o); }
  public boolean equals(Cantante c) {
    return nome.equals(c.nome);
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Cantante> s=new HashSet<Cantante>();
    Scanner scanner=new Scanner(System.in);
    Cantante c=new Cantante(scanner.nextLine());
    if(!s.add(c)) System.out.println("Cantante già esistente!");
    Canzone b=new Canzone(scanner.nextLine(), scanner.nextLine(),
        scanner.nextFloat());
    c.aggiungi(b);
    for(Cantante x: s) if(x.esegue(b)) System.out.println(x.getNome());
    String autore = scanner.nextLine();
    Cantante max = null;
    int maxC = 0;
    for(Cantante x: s) {
      int tot = x.autore(autore);
      if(tot >= maxC) { max = x; maxC = tot; }
    }
    System.out.println(max.getNome());
    int tot = 0;
    for(Cantante x: s) tot += x.autore(autore);
    System.out.println(tot);
  }
}
```