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

Appello del 13/7/2012

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

Esercizio 1 (4 punti)

1. Descrivere i tipi primitivi del linguaggio Java.
2. Discutere i diversi tipi di complessità computazionale di un algoritmo.

Esercizio 2 (2 punti)

Convertire in binario i numeri **-45** e **–77**, 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 N) {
  int sum=0;
  for(int i=0; i++<N;)
    sum+=V[i++];
  return sum;
}

public static int g(int V[],int N) {
  int j=0, sum=0;
  while(j++<N)
    sum+=f(V, j);
  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` dispari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (6 punti)

La Federazione Fans del Frisbee (FFF) si occupa dell'organizzazione di tornei di frisbee fra squadre dilettantistiche. A tal scopo, i dati relativi ad ogni squadra iscritta alla federazione vengono memorizzati registrando il nome della squadra ed il numero totale di partite vinte e perse (nel frisbee non esiste il pareggio). Si scriva una classe `Squadra` per la FFF che:

1. Possieda un opportuno costruttore con parametri (inizialmente una squadra non ha giocato alcuna partita).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Possieda il metodo `partiteGiocate` che, dato il numero di partite vinte e perse dalla squadra in un torneo, provveda ad aggiornare in maniera opportuna le variabili di istanza.
4. Presenti il metodo `toString` che fornisca una descrizione della squadra.
5. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Squadra` (l'uguaglianza va verificata unicamente sul nome della squadra).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Squadra` passato come parametro (la precedenza va data per ordine decrescente di vittorie e, in caso di parità, si procede per ordine alfabetico sul nome).

Esercizio 5 (7 punti)

Si scriva una classe `Torneo` che memorizzi le informazioni relative ai tornei organizzati dalla FFF. Per ogni torneo vengono memorizzate le squadre partecipanti (all'interno di un insieme), il luogo di svolgimento e la data di inizio del torneo stesso. La classe `Torneo` deve inoltre:

1. Presentare un opportuno costruttore con parametri (inizialmente, ad un torneo non è iscritta alcuna squadra).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del torneo (inclusa la descrizione di tutte le squadre partecipanti).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Squadra`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
5. Presentare il metodo `favorita` che restituisca la squadra che presenta più vittorie tra quelle iscritte al torneo.
6. Possedere il metodo `cerca` che, dato il nome di una squadra, indichi se tale squadra è iscritta al torneo o meno.

Esercizio 6 (8 punti)

Si scriva un'applicazione per la FFF che:

1. Crei una lista di oggetti `Torneo`.
2. Crei un oggetto `Torneo`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. in coda alla lista di cui al punto 1.
4. Crei un oggetto `Squadra`, lette da tastiera le informazioni necessarie.
5. Inserisca la squadra creata al punto 4. tra quelle iscritte al torneo di cui al punto 2.
6. Letto il numero di partite vinte e perse dalla squadra di cui al punto 4. nei vari tornei, provveda ad aggiornarne le statistiche.
7. Stampi a video il nome della squadra con più vittorie tra quelle iscritte nei tornei all'interno della lista di cui al punto 1.

Soluzione Esercizio 2

$-45_{10} = 11010011_2$
$-77_{10} = 10110011_2$

$10000110_2 = -122_{10}$

```
11010011+
10110011
--------
10000110
```

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`i++<N`	N/2 + 1	o (N+3)/2
`sum+=V[i++]`	N/2	o (N+1)/2
Totale	N + 3	o N + 4

Domanda 2:

2 assegnamenti	2
`while(j++<N)`	N + 1
`sum+=f(V, j)`	N
complessità di `f`	$N^2/2 + 4N + 1/2$
Totale	$N^2/2 + 6N + 7/2$

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

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

Soluzione Esercizio 4

```
class Squadra implements Comparable<Squadra> {
  private String nome;
  private int vittorie, sconfitte;

  public Squadra(String nome) {
    this.nome=nome;
    this.vittorie=0;
    this.sconfitte=0;
  }
  public String getNome() { return nome; }
  public int getVittorie() { return vittorie; }
  public int getSconfitte() { return sconfitte; }
  public void partiteGiocate(int vittorie, int sconfitte) {
    this.vittorie+=vittorie;
    this.sconfitte+=sconfitte; }
  public String toString() {
    return nome + ": " + vittorie + "/" + sconfitte;
  }
  public boolean equals(Object o) { return equals((Squadra) o); }
  public boolean equals(Squadra s) { return nome.equals(s.nome); }
  public int compareTo(Squadra s) {
    int ret=s.vittorie-this.vittorie;
    if(ret==0) ret=this.nome.compareTo(s.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;
class Torneo{
  private String luogo;
  private int g, m, a;
  private Set<Squadra> s;
  public Torneo(String luogo, int g, int m, int a) {
    this.luogo=luogo;
    this.g=g; this.m=m; this.a=a;
    s=new TreeSet<Squadra>();
  }
  public String getLuogo() { return luogo; }
  public String getData() { return g+"/"+m+"/"+a; }
  public String toString() {
    return luogo + ", " + getData() + ": " + s.toString();
  }
  public boolean aggiungi(Squadra s) { return this.s.add(s); }
  public Squadra favorita() {
    Squadra favorita=null;
    for(Squadra sq:s)
      if(favorita==null || sq.getVittorie()>favorita.getVittorie())
        favorita=sq;
    return favorita;
  }
  public boolean cerca(String nome) {
    for(Squadra sq:s) if(sq.getNome().equals(nome)) return true;
    return false;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;
class Applicazione {
  public static void main(String[] args) {
    List<Torneo> l=new LinkedList<Torneo>();
    Scanner scanner=new Scanner(System.in);
    Torneo t=new Torneo(scanner.nextLine(), scanner.nextInt(),
      scanner.nextInt(), scanner.nextInt());
    l.add(t);
    Squadra s=new Squadra(scanner.nextLine());
    if(!t.aggiungi(s)) System.out.println("Squadra già iscritta!");
    s.partiteGiocate(scanner.nextInt(), scanner.nextInt());
    Squadra favorita=null;
    for(Torneo x: l) {
      Squadra xf=x.favorita();
      if(favorita==null || xf.getVittorie()>favorita.getVittorie())
        favorita=xf;
    }
    if(favorita!=null) System.out.println(favorita.getNome());
  }
}
```