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

Appello del 21/10/2011

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

Esercizio 1 (4 punti)

1. Descrivere le principali differenze tra memoria principale e secondaria.
2. Definire il concetto di ricorsione.

Esercizio 2 (2 punti)

Convertire in binario i numeri **54** e **–76**, 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 i=0, sum=0;
  do
    sum+=V[i++];
  while(i++<M)
  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 `M` (si distinguano i casi in cui `M` 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 e si esprima `j=2 i`).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (5 punti)

In previsione delle prossime elezioni amministrative, il comune di Collate (Parma) deve approntare il sistema informativo per la gestione delle votazioni. A tal scopo, i dati relativi ad ogni candidato vengono memorizzati in un calcolatore, registrandone il nome, il cognome ed il numero di preferenze ricevute. Si scriva una classe `Candidato` per il comune di Collate (Parma) che:

1. Possieda un opportuno costruttore con parametri (inizialmente il candidato non ha alcun voto).
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Possieda il metodo `voto` che aggiunga un voto al candidato.
4. Presenti il metodo `toString` che fornisca una descrizione della fermata.
5. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Candidato` (l'uguaglianza va verificata solamente su nome e cognome).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Candidato` passato come parametro (la precedenza va data al candidato con più voti e, in caso di parità, si procede in ordine alfabetico per cognome e nome).

Esercizio 5 (8 punti)

Si scriva una classe `ListaElettorale` che memorizzi le informazioni relative alle liste elettorali. Per ogni lista vengono memorizzati il nome della lista ed il cognome del candidato sindaco. Inoltre, i candidati appartenenti alla lista vanno inseriti all'interno di una lista. La classe `ListaElettorale` deve inoltre:

1. Presentare un opportuno costruttore con parametri (inizialmente, una lista non contiene alcun candidato).
2. Presentare opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Possedere il metodo `toString` che fornisca la descrizione della lista (inclusa la descrizione di tutti i candidati).
4. Presentare il metodo `voto` che, dato il nome ed il cognome di un candidato, assegni il voto al candidato corrispondente, se questo esiste.
5. Possedere il metodo `aggiungi` che, dato un oggetto `Candidato`, lo inserisca in coda alla lista.
6. Presentare il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `ListaElettorale` (l'uguaglianza va verificata solamente sul nome della lista).
7. Possedere il metodo `capolista` che restituisca l'oggetto `Candidato` più votato della lista.
8. Presentare il metodo `votiTotali` che restituisca il numero totale di voti assegnati ai candidati della lista.

Esercizio 6 (7 punti)

Si scriva un'applicazione per il comune di Collate (Parma) che:

1. Crei un insieme di oggetti `ListaElettorale`.
2. Crei un oggetto `ListaElettorale`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. all'interno dell'insieme di cui al punto 1., verificando che tale insieme non contenga già un oggetto uguale.
4. Crei un oggetto `Candidato`, lette da tastiera le informazioni necessarie.
5. Inserisca il candidato creato al punto 4. nella lista di cui al punto 2.
6. Stampi a video la descrizione del candidato più votato in assoluto tra tutte le liste.

Soluzione Esercizio 2

$54_{10} = 00110110_2$
$-76_{10} = 10110100_2$

$11101010_2 = -22_{10}$

```
00110110+
10110100
11101010
```

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`++j<N`	`N`/2 + 1
`sum+=f(V, j++)`	`N`/2
complessità di `f`	$N^2/4 + 5\,N/2$
Totale	$N^2/4 + 7\,N/2 + 3$

complessità di `f`: $\sum_{\substack{j=2 \\ j \text{ pari}}}^{N} (j+4) = \sum_{i=1}^{\frac{N}{2}} (2i+4) = 2\frac{\frac{N}{2}\left(\frac{N}{2}+1\right)}{2} + \frac{4N}{2} = \frac{N^2}{4} + \frac{5N}{2}$

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

Soluzione Esercizio 4

```
class Candidato implements Comparable<Candidato> {
  private String nome, cognome;
  private int voti;

  public Candidato(String nome, String cognome) {
    this.nome=nome;
    this.cognome=cognome;
    this.voti=0;
  }
  public String getNome() { return nome; }
  public String getCognome() { return cognome; }
  public int getVoti() { return voti; }
  public void voto() { voti++; }
  public String toString() {
    return nome+ " " + cognome + ": " + voti;
  }
  public boolean equals(Object o) { return equals((Candidato) o); }
  public boolean equals(Candidato c) {
    return nome.equals(c.nome) && cognome.equals(c.cognome);
  }
  public int compareTo(Candidato c) {
    int ret=c.voti-this.voti;
    if(ret==0) ret=this.cognome.compareTo(c.cognome);
    if(ret==0) ret=this.nome.compareTo(c.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class ListaElettorale {
  private String nome, sindaco;
  private List<Candidato> l;

  public ListaElettorale(String nome, String sindaco) {
    this.nome=nome;
    this.sindaco=sindaco;
    l=new LinkedList< Candidato >();
  }
  public String getNome() { return nome; }
  public String getSindaco() { return sindaco; }
  public String toString() {
    return nome + " (" + sindaco + "):" + l.toString();
  }
  public void voto(String nome, String cognome) {
    for(Candidato c: l)
      if(c.getNome().equals(nome) && c.getCognome().equals(cognome))
        c.voto();
  }
  public void aggiungi(Candidato c) { l.add(c); }
  public boolean equals(Object o) { return equals((ListaElettorale) o); }
  public boolean equals(ListaElettorale l) { return nome.equals(l.nome); }
  public Candidato capolista() {
    Candidato max=null;
    for(Candidato c:l) if(max==null || c.getVoti()>max.getVoti()) max=c;
    return max;
  }
  public int votoTotali() {
    int totali=0;
    for(Candidato c:l) totali+=c.getVoti();
    return totali;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Set<ListaElettorale> s=new HashSet<ListaElettorale>();
    Scanner scanner=new Scanner(System.in);
    ListaElettorale l=new ListaElettorale(scanner.next(), scanner.next());
    if(!s.add(l)) System.out.println("Lista già presente!");
    Candidato c=new Candidato(scanner.next(), scanner.next());
    l.aggiungi(c);
    Candidato max=null;
    for(ListaElettorale x: s) {
      Candidato cl=x.capolista();
      if(cl!=null && (max==null || cl.getVoti>max.getVoti()))
        max=cl;
    }
    System.out.println(max);
  }
}
```