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

Appello del 22/10/2010

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

Esercizio 1 (4 punti)

1. Descrivere le modalità di studio della complessità temporale di un algoritmo.
2. Si illustri il concetto di ricorsione *tail*, indicando quali siano le differenze nell'utilizzo della memoria rispetto ad una normale ricorsione lineare.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **0,008** supponendo di utilizzare 8 bit per la mantissa (in modulo e segno) ed 8 bit per l'esponente (in complemento a 2).

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[], int N, int M) {
  int sum=0;
  for(i=M-1; i>N-1; i--)
    sum+=V[i];
  return sum;
}

public static int g(int V[],int N) {
  int j, sum=0;
  for(j=1; j<N; j+=2)
    sum+=f(V, --j, N);
  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 minori di `N` da quelli in cui assume valori maggiori o uguali a `N`).
2. Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N`>0).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (5 punti)

In previsione dell'inizio della stagione invernale, l'agenzia di turismo della località montana Campinove ha deciso di informatizzare la gestione degli impianti di risalita presenti sul territorio comunale. Per ogni impianto vanno memorizzati il nome, la lunghezza (in metri) ed il dislivello (sempre in metri). Si scriva una classe `Impianto` per la località montana Campinove 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 la descrizione dell'impianto.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Impianto` (l'uguaglianza va verificata unicamente sul nome dell'impianto).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Impianto` passato come parametro (la precedenza va verificata per lunghezza decrescente e, in caso di parità, si procede per ordine alfabetico sul nome).

Esercizio 5 (7 punti)

Si scriva una classe `Comprensorio` che memorizzi le informazioni relative agli impianti inclusi all'interno di un determinato comprensorio sciistico. Oltre al nome del comprensorio, i vari impianti vanno inseriti all'interno di una lista. La classe `Comprensorio` deve inoltre:

1. Presentare un opportuno costruttore (inizialmente, la lista di impianti è vuota).
2. Possedere il metodo `toString` che fornisca la descrizione del comprensorio (inclusa la descrizione di tutti gli impianti ivi situati).
3. Possedere il metodo `aggiungi` che, dato un oggetto `Impianto`, lo inserisca all'interno della lista, mantenendo tale lista ordinata secondo il punto 5. dell'esercizio 4 (suggerimento: si utilizzi il metodo `add(int i, Impianto imp)` della classe `List<Impianto>`).
4. Presentare il metodo `cerca` che, dato il nome di un impianto, restituisca l'impianto più lungo all'interno della lista avente tale nome.

Esercizio 6 (8 punti)

Si scriva un'applicazione per la località montana Campinove che:

1. Crei un insieme di oggetti `Comprensorio`.
2. Crei un oggetto `Comprensorio`, 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 `Impianto`, lette da tastiera le informazioni necessarie, e lo inserisca tra gli impianti del comprensorio di cui al punto 2.
5. Letto da tastiera il nome di un impianto, stampi a video la descrizione di tutti i comprensori che includono un impianto con tale nome.
6. Tra tutti i comprensori di cui al punto 5., stampi a video la descrizione del comprensorio che include l'impianto con il nome letto al punto 5. avente la lunghezza maggiore.

Soluzione Esercizio 2

$0{,}008_{10} = -0{,}0000001000001_2$ quindi la mantissa è **(0)1000001** e l'esponente -6_{10} = **11111010**.

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`j<N`	N
`sum+=f(V, --j, N)`	N – 1
`j+=2`	N – 1
complessità di `f`	$3N^2/2 + 9N/2 - 6$
Totale	$3N^2/2 + 15N/2 - 6$

complessità di `f`: $\sum_{j=0}^{N-2}(3N-3j+3) = 3N(N-1) - \frac{3}{2}(N-1)(N-2) + 3(N-1) = 3\frac{N^2}{2} + 9\frac{N}{2} - 6$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 4

```
class Impianto implements Comparable<Impianto> {
  private String nome;
  private int lunghezza, dislivello;

  public Impianto(String nome, int lunghezza, int dislivello) {
    this.nome=nome;
    this.lunghezza=lunghezza;
    this.dislivello=dislivello;
  }

  public String getNome() { return nome; }
  public int getLunghezza() { return lunghezza; }
  public int getDislivello() { return dislivello; }

  public String toString() {
    return nome + "(" + lunghezza + "): " + dislivello;
  }

  public boolean equals(Object o) { return equals((Impianto) o); }
  public boolean equals(Impianto i) {
    return nome.equals(i.nome);
  }

  public int compareTo(Impianto i) {
    int ret=i.lunghezza -this.lunghezza;
    if(ret==0) ret=this.nome.compareTo(i.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Comprensorio {
  private String nome;
  private List<Impianto> l;

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

  public String toString() { return nome + ":\n" + l; }

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

  public Impianto cerca(String nome) {
    for(Impianto i:l) if(i.getNome().equals(nome)) return i;
    return null;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Comprensorio> s=new HashSet<Comprensorio>(); // domanda 1
    Scanner scanner=new Scanner(System.in);
    Comprensorio c=new Comprensorio(scanner.next()); // domanda 2
    if(!s.add(c)) System.out.println("Comprensorio già esistente!");
      // domanda 3
    c.aggiungi(new Impianto(scanner.next(), scanner.nextInt(),
      scanner.nextInt())); // domanda 4
    String nome=scanner.next();
    Impianto lungo=null;
    int max=0;
    for(Comprensorio cp:s) {
      Impianto i=cp.cerca(nome);
      if(i!=null) {
        System.out.println(cp);
        if(i.getLunghezza()>max) {
          lungo=i;
          max=i.getLunghezza();
        }
      }
    }
    if(lungo!=null) System.out.println(lungo);
    else System.out.println("Nessun impianto con tale nome!"); // domanda 5
  }
}
```