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

Appello del 10/9/2010

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. Illustrare la sequenza di operazioni elementari necessarie per l'esecuzione dell'istruzione `STORE1 Y` (con `Y` indirizzo di memoria) all'interno della macchina di Von Neumann.

Esercizio 2 (2 punti)

Convertire in binario i numeri **39** 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.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[],int M) {
  int sum=0, i=0;
  for(; i<2*M; ++i)
    sum+=V[i++];
  return sum;
}

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

1. Calcolare la complessità in passi base del metodo `f` nei termini del parametro `M` (si supponga `M` 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` + 1).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 4 (5 punti)

L'agenzia di viaggi "Sedano" ha deciso di informatizzare la gestione degli itinerari proposti ai turisti che si recano a visitare lo stato caraibico di St. Marquez. Innanzitutto, vanno memorizzati i dati relativi ad ogni località turistica, ovvero il nome della località e la durata (in giorni) ed il costo (in euro) di una visita. Si scriva una classe `Localita` per l'agenzia "Sedano" 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 della località.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Localita` (l'uguaglianza va verificata unicamente sul nome della località).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Localita` passato come parametro (la precedenza va verificata per durata decrescente e, in caso di parità, si procede per ordine alfabetico sul nome).

Esercizio 5 (8 punti)

Si scriva una classe `Itinerario` che memorizzi le informazioni relative alle località incluse in un itinerario proposto dall'agenzia. Oltre al nome dell'itinerario ed al numero massimo di partecipanti, le varie località vanno inserite all'interno di un insieme. La classe `Itinerario` deve inoltre:

1. Presentare un opportuno costruttore (inizialmente, l'itinerario non contiene alcuna località).
2. Presentare un metodo che restituisca il nome dell'itinerario.
3. Possedere il metodo `toString` che fornisca la descrizione dell'itinerario (inclusa la descrizione di tutte le località visitate).
4. Presentare il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Itinerario` (l'uguaglianza va verificata sul nome dell'itinerario).
5. Possedere il metodo `aggiungi` che, dato un oggetto `Localita`, lo inserisca all'interno dell'insieme, controllando che questo non contenga già un oggetto uguale.
6. Presentare il metodo `cerca` che, dato il nome di una località, indichi se l'itinerario comprende o meno una visita a tale località.
7. Possedere il metodo `costo` che restituisca il costo totale della visita a tutte le località incluse.

Esercizio 6 (7 punti)

Si scriva un'applicazione per l'agenzia "Sedano" che:

1. Crei una lista di oggetti `Itinerario`.
2. Crei un oggetto `Itinerario`, 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 `Localita`, lette da tastiera le informazioni necessarie, e lo inserisca tra le località visitabili nell'itinerario di cui al punto 2.
5. Letto da tastiera il nome di una località stampi a video il nome dell'itinerario a costo minore che permette di visitare la località richiesta; si tenga in considerazione anche il caso in cui nessun itinerario comprenda tale località.

Soluzione Esercizio 2

$39_{10} = 00100111_2$
$-77_{10} = 10110011_2$

$11011010_2 = -38_{10}$

```
00100111+
10110011
11011010
```

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2
`i<2*M`	`M` + 1
`sum+=V[i++]`	`M`
`++i`	`M`
Totale	3`M` + 3

Domanda 2:

2 assegnamenti	2
`j<N`	`N`/2 + 1
`sum+=f(V, ++j)`	`N`/2
`j++`	`N`/2
complessità di `f`	3`N`^{2}/4 + 3`N`/2
Totale	3`N`^{2}/4 + 3`N` + 3

complessità di `f`: $\sum_{\substack{j=1 \\ (j \text{ dispari})}}^{N-1}(3j+3) = \sum_{i=0}^{N/2-1}(6i+6) = 3\frac{N}{2}\left(\frac{N}{2}-1\right)+3N = 3\frac{N^2}{4}+3\frac{N}{2}$

Domanda 3:
Complessità asintotica: $O(\mathtt{N}^2)$

Soluzione Esercizio 4

```
class Localita implements Comparable<Localita> {
  private String nome;
  private int durata, costo;

  public Localita (String nome, int durata, int costo) {
    this.nome=nome;
    this.durata=durata;
    this.costo=costo;
  }

  public String getNome() { return nome; }
  public int getDurata() { return durata; }
  public int getCosto() { return costo; }

  public String toString() {
    return nome + "(" + durata + "): " + costo;
  }

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

  public int compareTo(Localita l) {
    int ret=l.durata-this.durata;
    if(ret==0) ret=this.nome.compareTo(l.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Itinerario {
  private String nome;
  private int partecipanti;
  private Set<Localita> s;

  public Itinerario(String nome, int partecipanti) {
    this.nome=nome;
    this.partecipanti=partecipanti;
    s=new TreeSet<Localita>();
  }

  public String getNome() { return nome; }
  public String toString() { return nome+"("+partecipanti+"):\n"+s; }

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

  public boolean aggiungi(Localita l) { return s.add(l); }

  public boolean cerca(String nome) {
    for(Localita l:s) if(l.getNome().equals(nome)) return true;
    return false;
  }

  public int costo() {
    int tot=0;
    for(Localita l:s) tot+=l.getCosto();
    return tot;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Itinerario> l=new ArrayList<Itinerario>(); // domanda 1
    Scanner scanner=new Scanner(System.in);
    Itinerario i=new Itinerario(scanner.next(), scanner.nextInt());
      // domanda 2
    l.add(i); // domanda 3
    i.aggiungi(new Localita(scanner.next(), scanner.nextInt(),
      scanner.nextInt())); // domanda 4
    String nome=scanner.next();
    Itinerario migliore=null;
    int min=0;
    for(Itinerario it:l)
      if(it.cerca(nome)) {
        int costo=it.costo();
        if(migliore==null||costo<min) {
          migliore=it;
          min=costo;
        }
      }
    if(migliore!=null) System.out.println(migliore.getNome());
    else System.out.println("Località non visitabile!"); // domanda 5
  }
}
```