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

Appello del 8/7/2016

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. Definire il concetto di ricorsione.

Esercizio 2 (2 punti)

Convertire in binario i numeri **77** e **53**, 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 i=N/2, sum=0;
  while(i-->0)
    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 `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 (5 punti)

Pino (Pinuccio per gli amici) Damiani, ingegnere delle telecomunicazioni, è manager dei sistemi d'antenna di una società di telefonia cellulare. Per la gestione degli impianti, la società ha deciso di affidare a Pinuccio la memorizzazione delle informazioni relative alle antenne dei comuni della provincia di Bologna. In particolare, per ogni impianto occorre registrare l'indirizzo, la matricola e la potenza in Watt. Si scriva una classe `Antenna` per Pinuccio Damiani che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca una descrizione dell'antenna.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Antenna` (la verifica va fatta unicamente sull'indirizzo).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Antenna` passato come parametro (in ordine alfabetico per indirizzo e, a parità, per matricola decrescente).

Esercizio 5 (8 punti)

Si scriva una classe `Comune` che memorizzi le informazioni riguardanti le antenne presenti in un comune della provincia. Per ogni comune occorre memorizzare il nome e il CAP, mentre le antenne vanno inserite all'interno di una lista. La classe `Comune` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, il comune non contiene antenne).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del comune (inclusa la descrizione di tutte le antenne).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Comune` (la verifica va effettuata unicamente sul CAP).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Antenna`, lo inserisca all'interno della lista, mantenendo quest'ultima ordinata secondo il punto 5. dell'Esercizio 4.
6. Possedere il metodo `potenza` che restituisca la potenza totale di tutte le antenne del comune.
7. Presentare il metodo `indirizzo` che, dato un indirizzo, indichi se esiste o meno un'antenna in tale indirizzo.

Esercizio 6 (8 punti)

Si scriva un'applicazione per Pinuccio Damiani che:

1. Crei un insieme di oggetti `Comune`.
2. Crei un oggetto `Comune`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2all'interno dell'insieme di cui al punto 1., controllando che tale inserimento sia possibile.
4. Crei un oggetto `Antenna`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno del comune di cui al punto 2.
6. Letto da tastiera un CAP, stampi a video la potenza totale di tutte le antenne presenti nel comune avente tale CAP.
7. Stampi a video il nome del comune che presenta la potenza totale maggiore tra quelli dell'insieme di cui al punto 1.

Soluzione Esercizio 2

$77_{10} = 01001101_2$
$53_{10} = 00110101_2$

$$\begin{array}{r} 01001101+ \\ \underline{00110101} \\ 10000010 \end{array}$$

$10000010_2 = -126_{10}$

Il numero rappresentato è diverso dall'originale (130) poiché quest'ultimo non è rappresentabile con soli 8 bit.

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`j<N`	(N + 1)/2 + 1
`sum+=f(V, --j)`	(N + 1)/2
`j>0`	(N + 1)/2
complessità di `f`	$N^2/4 + 3N/2 + 5/4$
Totale	$N^2/4 + 3N + 23/4$

complessità. di `f`: $\sum_{\substack{j=1 \\ j\ dispari}}^{N} (j+2) = \sum_{i=0}^{\frac{N-1}{2}} (2i+3) = \frac{(N+1)}{2}\frac{(N-1)}{2} + 3\frac{N+1}{2} = \frac{N^2}{4} + \frac{3N}{2} + \frac{5}{4}$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 4

```
class Antenna implements Comparable<Antenna> {
  private String indirizzo;
  private int matr, potenza;

  public Antenna(String indirizzo, int matr, int potenza) {
    this.indirizzo = indirizzo;
    this.matr = matr;
    this.potenza = potenza;
  }

  public String getIndirizzo() { return indirizzo; }
  public int getMatricola() { return matr; }
  public int getPotenza() { return potenza; }

  public String toString() {
    return indirizzo + ": " + matr + " (" + potenza +")";
  }

  public boolean equals(Object o) { return equals((Antenna) o); }
  public boolean equals(Antenna a) { return indirizzo.equals(a.indirizzo); }

  public int compareTo(Antenna a) {
    int ret = this.indirizzo.compareTo(a.indirizzo);
    if(ret==0) ret = a.matr - this.matr;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Comune {
  private List<Antenna> l;
  private String nome, cap;
  public Comune(String nome, String cap) {
    this.nome = nome;
    this.cap = cap;
    l = new LinkedList<Antenna>();
  }
  public String getNome() { return nome; }
  public String getCAP() { return cap; }
  public String toString() {
    return nome + "(" + cap+ "):" + l;
  }
  public boolean equals(Object o) { return equals((Comune) o); }
  public boolean equals(Comune c){ return cap.equals(c.cap); }
  public void aggiungi(Antenna a) {
    int i = 0;
    while((i<l.size())&&(l.get(i).compareTo(a)<0)) i++;
    l.add(i, a);
  }
  public int potenza() {
    int potenza=0;
    for(Antenna a: l) potenza+=a.getPotenza();
    return potenza;
  }
  public boolean indirizzo(String indirizzo) {
    for(Antenna a: l)
      if(a.getIndirizzo().equals(indirizzo)) return true;
    return false;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Comune> s = new HashSet<Comune>();
    Scanner scanner = new Scanner(System.in);
    Comune c = new Comune(scanner.nextLine(), scanner.next());
    if(!s.add(c)) System.out.println("Comune già presente!");
    Antenna a = new Antenna(scanner.nextLine(), scanner.nextInt(),
        scanner.nextInt());
    c.aggiungi(a);
    String cap = scanner.nextLine();
    for(Comune x: s)
      if(x.getCAP().equals(cap)) {
        System.out.println(x.potenza()); break; }
    int potenza = 0;
    Comune max=null;
    for(Comune x: s) {
      int xp=x.potenza();
      if(max==null||xp>potenza) { max=x; potenza=xp; }
    }
    System.out.println(max.getNome());
  }
}
```