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

Appello del 15/6/2018

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

Esercizio 1 (4 punti)

1. Illustrare i componenti interni di una CPU.
2. Descrivere il contenuto e il funzionamento dei record di attivazione nella JVM.

Esercizio 2 (2 punti)

Rappresentare in binario il numero **53,75** supponendo di utilizzare 8 bit per la mantissa (in modulo e segno) ed 8 bit per l'esponente (in complemento a 2). Si riconverta infine il numero rappresentato in base 10, motivando eventuali differenze con il numero originale.

Esercizio 3 (5 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[],int N) {
  int i=N, sum=0;
  while(i-->0)
    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 (5 punti)

Enrico Pupillo ha finalmente comprato casa e, a lavori ormai ultimati, sta programmando il trasloco di mobili e proprietà. Per questo ha deciso di informatizzare lo spostamento degli scatoloni tra la vecchia e la nuova magione. In particolare, per ogni scatolone occorre memorizzare la descrizione del contenuto (es., "piatti", "posate", "libri"), il peso in Kg e un numero identificativo. Si scriva una classe `Scatolone` per Enrico Pupillo 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 dello scatolone.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Scatolone` (la verifica va fatta sul numero identificativo).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Scatolone` passato come parametro (per ordine crescente di peso e, a parità, per numero crescente).

Esercizio 5 (8 punti)

Si scriva una classe `Camion` che registri le informazioni riguardanti gli scatoloni caricati su un certo camion. Per ogni camion occorre memorizzare la targa e la capienza (in quintali), mentre gli scatoloni vanno memorizzati all'interno di un insieme (il cui peso totale non deve superare la capienza). La classe `Camion` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, l'insieme degli scatoloni è vuoto).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del camion (inclusa la descrizione di tutti gli scatoloni ivi caricati).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Camion` (la verifica va effettuata unicamente sulla targa).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Scatolone`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `contenuto` che, data una stringa, restituisca un insieme contenente tutti gli scatoloni aventi tale contenuto caricati nel camion.
7. Possedere il metodo `peso` che restituisca il peso totale degli scatoloni caricati sul camion.

Esercizio 6 (8 punti)

Si scriva un'applicazione per Enrico Pupillo che:

1. Crei una lista di oggetti `Camion`.
2. Crei un oggetto `Camion`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. in testa alla lista di cui al punto 1.
4. Crei un oggetto `Scatolone`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. tra quelli contenuti nel camion di cui al punto 2., controllando che tale inserimento sia possibile.
6. Letto da tastiera la descrizione di un contenuto, stampi a video la targa di tutti i camion che non caricano scatoloni con tale contenuto.
7. Stampi a video la descrizione del camion avente la capienza residua massima tra quelli della lista di cui al punto 1.

Soluzione Esercizio 2

$53{,}75_{10} = 110101{,}11_2$ quindi la mantissa è **(0)1101011**, l'esponente 6_{10} = **00000110**.
Il numero rappresentato è 53,5, diverso dall'originale poiché quest'ultimo possiede un numero di cifre significative maggiore di 7.

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`i-->0`	N/2 + 1	o (N + 1)/2 + 1
`sum+=V[--i]`	N/2	o (N + 1)/2
Totale	N + 3	o N + 4

Domanda 2:

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

complessità. di `f`: $\sum_{\substack{j=1 \\ j \text{ dispari}}}^{N-2} (j+4) = \sum_{i=0}^{\frac{N-3}{2}} (2i+5) = \frac{(N-1)}{2}\frac{(N-3)}{2} + 5\frac{N-1}{2} = \frac{N^2}{4} + \frac{3N}{2} - \frac{7}{4}$

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

Soluzione Esercizio 4

```
class Scatolone implements Comparable<Scatolone> {
  private int numero;
  private float peso;
  private String contenuto;

  public Scatolone(int numero, String contenuto, float peso) {
    this.numero = numero;
    this.contenuto = contenuto;
    this.peso = peso;
  }

  public int getNumero() { return numero; }
  public String getContenuto() { return contenuto; }
  public float getPeso() { return peso; }

  public String toString() {
    return numero + " (" + contenuto + "): " + peso;
  }

  public boolean equals(Object o) { return equals((Scatolone) o); }
  public boolean equals(Scatolone s) {
    return numero==s.numero;
  }

  public int compareTo(Scatolone s) {
    if(this.peso<s.peso) return -1;
    if(this.peso>s.peso) return 1;
    return this.numero - s.numero;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Camion {
  private String targa;
  private float capienza;
  private Set<Scatolone> s;

  public Camion(String targa, int capienza) {
    this.targa = targa;
    this.capienza = capienza;
    s = new HashSet<Scatolone>();
  }
  public String getTarga() { return targa; }
  public float getCapienza() { return capienza; }

  public String toString() { return targa + " (" + capienza + "): " + s;}
  public boolean equals(Object o) { return equals((Camion) o); }
  public boolean equals(Camion c){ return this.targa.equals(c.targa); }

  public boolean aggiungi(Scatolone s) {
    if(peso()+s.getPeso()<capienza*100) return this.s.add(s);
    return false;
  }

  public Set<Scatolone> contenuto(String s) {
    Set<Scatolone> result = new TreeSet<Scatolone>();
    for(Scatolone x: this.s) if(x.getContenuto().equals(s)) result.add(x);
    return result;
  }

  public float peso() {
    float peso=0;
    for(Scatolone x: s) peso+=x.getPeso();
    return peso;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Camion> l = new ArrayList<Camion>();
    Scanner scanner = new Scanner(System.in);
    Camion c = new Camion(scanner.next(), scanner.nextInt());
    l.add(0,c);
    Scatolone s = new Scatolone(scanner.nextInt(), scanner.nextLine(),
        scanner.nextFloat());
    if(!c.aggiungi(s)) System.out.println("Inserimento non avvenuto!");
    String contenuto=scanner.nextLine();
    for(Camion x: l)
      if(x.contenuto(contenuto).isEmpty()) System.out.println(x);
    Camion max = null;
    float maxCapienza = 0;
    for(Camion x: l) {
      float capienza=x.getCapienza()*100-x.peso();
      if(capienza>maxCapienza) { max=x; maxCapienza=capienza; }
    }
    System.out.println(max.toString());
  }
}
```