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

Appello del 18/7/2019

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. 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 **65,5** 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 M, int N) {
  int i=M, sum=0;
  do
    sum+=V[i];
  while(++i<N);
  return sum;
}

public static int g(int V[], int M, int N) {
  int j, sum;
  for(j=0, sum=0; ++j<M; j++)
    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).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `M` e `N` (si supponga `M` pari e minore di `N`).
3. Calcolare la complessità asintotica del metodo `g` nei termini dei parametri `M` e `N`.

Esercizio 4 (5 punti)

Dopo un anno passato in una compagine minore, Matteo Contini e Samuele Corretti sono finalmente giunti in una squadra vincente. Purtroppo, il campionato di Matteo è stato reso difficile dal protrarsi dei problemi di stabilità al ginocchio; per questo, ha chiesto a Samuele, laureato magistrale in Scienze Motorie, di preparare una serie di allenamenti di potenziamento muscolare da svolgere in palestra. Samuele ha perciò deciso di informatizzare la gestione degli allenamenti di Matteo. Per questo, per ogni esercizio occorre memorizzare il nome, il numero di Kg da sollevare e il numero di ripetizioni da eseguire. Si scriva una classe `Esercizio` per Matteo e Samuele 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'esercizio.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Esercizio` (la verifica va fatta su nome e peso).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Esercizio` passato come parametro (in ordine alfabetico per nome e, a parità, per peso crescente).

Esercizio 5 (8 punti)

Si scriva una classe `Scheda` che registri le informazioni riguardanti le schede di esercizi che verranno effettuate in un dato allenamento giornaliero. Per ogni scheda occorre memorizzare il giorno della settimana e la durata prevista (in minuti), mentre gli esercizi vanno inseriti all'interno di un insieme. La classe `Scheda` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, l'insieme di esercizi 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 della scheda (inclusa la descrizione di tutti gli esercizi).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Scheda` (la verifica va effettuata sul giorno della settimana).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Esercizio`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `esercizio` che, dato un nome, restituisca il numero di esercizi con tale nome presenti all'interno della scheda.
7. Possedere il metodo `chilogrammi` che restituisca il peso totale sollevato negli esercizi della scheda.

Esercizio 6 (8 punti)

Si scriva un'applicazione per Matteo e Samuele che:

1. Crei una lista di oggetti `Scheda`.
2. Crei un oggetto `Scheda`, 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 `Esercizio`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno della scheda di cui al punto 2., controllando che tale inserimento sia possibile.
6. Stampi a video il giorno delle schede che non prevedono esercizi di tipo "squat".
7. Stampi a video la descrizione della scheda che prevede il maggior peso totale sollevato.

Soluzione Esercizio 2

$65{,}5_{10} = 1000001{,}1_2$ quindi la mantissa è **(0)1000001**, l'esponente 7_{10} = **00000111**.
Il numero rappresentato è 65, diverso dall'originale poiché quest'ultimo ha un numero di cifre significative maggiore di 7.

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`sum+=V[i]`	$N - M$	o 1
`++i<N`	$N - M$	o 1
Totale	$2N - 2M + 2$	o 4

Domanda 2:

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

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

$$MN - \frac{M^2}{2} + M$$

Domanda 3:
Complessità asintotica: $O(MN)$

Soluzione Esercizio 4

```
class Esercizio implements Comparable<Esercizio> {
  private String nome;
  private int peso, ripetizioni;

  public Esercizio(String nome, int peso, int ripetizioni) {
    this.nome = nome;
    this.peso = peso;
    this.ripetizioni = ripetizioni;
  }

  public String getNome() { return nome; }
  public int getPeso() { return peso; }
  public int getRipetizioni() { return ripetizioni; }

  public String toString() {
    return nome + " (" + peso + "kg): " + ripetizioni;
  }

  public boolean equals(Object o) { return equals((Esercizio) o); }
  public boolean equals(Esercizio e) {
    return nome.equals(e.nome) && peso==e.peso;
  }

  public int compareTo(Esercizio e) {
    int ret = this.nome.compareTo(e.nome);
    if(ret==0) ret = this.peso-e.peso;
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Scheda {
  private String giorno;
  int durata;
  private Set<Esercizio> s;

  public Scheda(String giorno, int durata) {
    this.giorno = giorno;
    this.durata = durata;
    s = new HashSet<Esercizio>();
  }

  public String getGiorno() { return giorno; }
  public int getDurata() { return durata; }
  public String toString() { return giorno + "(" + durata+ "): " + s;}
  public boolean equals(Object o) { return equals((Scheda) o); }
  public boolean equals(Scheda s){ return giorno.equals(s.giorno); }

  public boolean aggiungi(Esercizio e) { return s.add(e); }

  public int esercizio(String nome) {
    int result=0;
    for(Esercizio e: s) if(e.getNome().equals(nome)) result++;
    return result;
  }

  public int chilogrammi() {
    int result=0;
    for(Esercizio e: s) result+=e.getPeso()*e.getRipetizioni();
    return result;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Scheda> l = new LinkedList<Scheda>();
    Scanner scanner = new Scanner(System.in);
    Scheda s = new Scheda(scanner.next(), scanner.nextInt());
    l.add(0,s);
    Esercizio e = new Esercizio (scanner.nextLine(), scanner.nextInt(),
        scanner.nextInt());
    if(!s.aggiungi(e)) System.out.println("Inserimento non avvenuto!");
    for(Scheda x: l)
      if(x.esercizio("squat")==0) System.out.println(x.getGiorno());
    Scheda max = null;
    int maxKG = 0;
    for(Scheda x: l) {
      int kg = x.chilogrammi();
      if(kg > maxKG) { max=x; maxKG=kg; }
    }
    System.out.println(max);
  }
}
```