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

Appello del 14/10/2016

Compito A

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

Esercizio 1 (4 punti)

1. Illustrare le fasi di realizzazione di un programma Java (compresa l'esecuzione del programma stesso).
2. Descrivere le modalità di studio della complessità temporale di un algoritmo.

Esercizio 2 (2 punti)

Convertire in binario i numeri **44** e **85**, 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=0, sum=0;
  while(i++<N)
    sum+=V[++i];
  return sum;
}

public static int g(int V[], int N) {
  int j=0, sum=0;
  do
    sum+=f(V, ++j);
  while(j<N);
  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)

Mattia è il rampollo della famiglia De Gianni, proprietaria della famosa catena di gelaterie "Simoni". Visto il recente ampliamento della catena, con diversi negozi in centro città, Mattia ha deciso di gestire all'interno di un calcolatore le informazioni relative ai gusti e agli ingredienti dei gelati proposti. In particolare, per ogni ingrediente occorre registrare il nome e il numero di calorie all'etto. Si scriva una classe `Ingrediente` per Mattia De Gianni 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'ingrediente.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Ingrediente` (la verifica va fatta esclusivamente sul nome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Ingrediente` passato come parametro (in ordine decrescente per calorie e, a parità, per ordine alfabetico del nome).

Esercizio 5 (7 punti)

Si scriva una classe `Gusto` che memorizzi le informazioni riguardanti un gusto di gelato creato per le gelaterie "Simone". Per ogni gusto occorre memorizzare il nome e il numero di barattoli prodotti, mentre gli ingredienti vanno inseriti all'interno di un insieme. La classe `Gusto` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, il gusto non ha ingredienti).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del gusto (inclusa la descrizione di tutti gli ingredienti contenuti).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Gusto` (la verifica va effettuata esclusivamente sul nome).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Ingrediente`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `ingrediente` che, dato il nome di un ingrediente, indichi se tale ingrediente è contenuto o meno nel gusto a disposizione.

Esercizio 6 (7 punti)

Si scriva un'applicazione per Mattia De Gianni che:

1. Crei una lista di oggetti `Gusto`.
2. Crei un oggetto `Gusto`, 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 `Ingrediente`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno del gusto di cui al punto 2., controllando che tale inserimento sia possibile.
6. Letto da tastiera il nome di un ingrediente, stampi a video il nome dei gusti che non contengono tale ingrediente all'interno della lista di cui al punto 1.
7. Letto da tastiera il nome di un ingrediente, stampi a video il numero totale di barattoli dei gusti che contengono tale ingrediente all'interno della lista di cui al punto 1.

Soluzione Esercizio 2

$44_{10} = 00101100_2$
$85_{10} = 01010101_2$

$$\begin{array}{r} 00101100+ \\ \underline{01010101} \\ 10000001 \end{array}$$

$10000001_2 = -127_{10}$

Il numero rappresentato è diverso dall'originale (129) poiché quest'ultimo non è rappresentabile con soli 8 bit.

Soluzione Esercizio 3

Domanda 1:

2 assegnamenti	2	o 2
`i++<N`	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
`sum+=f(V, ++j)`	N
`j<N`	N
complessità di `f`	$N^2/2 + 4N + 1/2$
Totale	$N^2/2 + 6N + 5/2$

complessità. di `f`: $\sum_{\substack{j=1 \\ j \text{ pari}}}^{N} (j+3) + \sum_{\substack{j=1 \\ j \text{ dispari}}}^{N} (j+4) = \sum_{j=1}^{N}(j+3) + \sum_{\substack{j=1 \\ j \text{ dispari}}}^{N} 1 =$

$$= \frac{N(N+1)}{2} + 3N + \frac{N+1}{2} = \frac{N^2}{2} + 4N + \frac{1}{2}$$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 4

```
class Ingrediente implements Comparable<Ingrediente> {
  private String nome;
  private int calorie;

  public Ingrediente(String nome, int calorie) {
    this.nome = nome;
    this.calorie = calorie;
  }

  public String getNome() { return nome; }
  public int getCalorie() { return calorie; }

  public String toString() {
    return nome + " (" + calorie + ")";
  }

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

  public int compareTo(Ingrediente i) {
    int ret = i.calorie - calorie;
    if(ret==0) ret = nome.compareTo(i.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Gusto {
  private String nome;
  private int barattoli;
  private Set<Ingrediente> s;
  public Gusto(String nome, int barattoli) {
    this.nome = nome;
    this.barattoli = barattoli;
    s = new HashSet< Ingrediente >();
  }

  public String getNome() { return nome; }
  public int getBarattoli() { return barattoli; }
  public String toString() {
    return nome+ "(" + barattoli + "): " + s;
  }

  public boolean equals(Object o) { return equals((Gusto) o); }
  public boolean equals(Gusto g){ return nome.equals(g.nome); }

  public boolean aggiungi(Ingrediente i) { return s.add(i); }
  public boolean ingrediente(String s) {
    for(Ingrediente i: this.s) if(i.getNome().equals(s)) return true;
    return false;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    List<Gusto> l = new ArrayList<Gusto>();
    Scanner scanner = new Scanner(System.in);
    Gusto g = new Gusto(scanner.nextLine(), scanner.nextInt());
    l.add(g);
    Ingrediente i = new Ingrediente(scanner.nextLine(),
      scanner.nextInt());
    if(!g.aggiungi(i)) System.out.println("Ingrediente già esistente!");
    String nome = scanner.nextLine();
    for(Gusto x: l)
      if(!x.ingrediente(nome)) System.out.println(x.getNome());
    nome = scanner.nextLine();
    int totale = 0;
    for(Gusto x: l)
      if(x.ingrediente(nome)) totale+=x.getBarattoli();
    System.out.println(totale);
  }
}
```

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

Appello del 14/10/2016

Compito B

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

Esercizio 1 (4 punti)

1. Illustrare le fasi di realizzazione di un programma Java (compresa l'esecuzione del programma stesso).
2. Descrivere le modalità di studio della complessità spaziale di un algoritmo.

Esercizio 2 (2 punti)

Convertire in binario i numeri **–44** e **–85**, 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=0, sum=0;
  while(++i<N)
    sum+=V[i++];
  return sum;
}

public static int g(int V[], int N) {
  int j=0, sum=0;
  do
    sum+=f(V, j++);
  while(j<N);
  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)

Mattia è il rampollo della famiglia De Gianni, proprietaria della famosa catena di gelaterie "Simoni". Visto il recente ampliamento della catena, con diversi negozi in centro città, Mattia ha deciso di gestire all'interno di un calcolatore le informazioni relative ai gusti e agli ingredienti dei gelati proposti. In particolare, per ogni ingrediente occorre registrare il nome e il numero di calorie all'etto. Si scriva una classe `Ingrediente` per Mattia De Gianni 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'ingrediente.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Ingrediente` (la verifica va fatta esclusivamente sul nome).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Ingrediente` passato come parametro (in ordine crescente per calorie e, a parità, per ordine alfabetico del nome).

Esercizio 5 (7 punti)

Si scriva una classe `Gusto` che memorizzi le informazioni riguardanti un gusto di gelato creato per le gelaterie "Simone". Per ogni gusto occorre memorizzare il nome e il numero di barattoli prodotti, mentre gli ingredienti vanno inseriti all'interno di una lista. La classe `Gusto` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, il gusto non ha ingredienti).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del gusto (inclusa la descrizione di tutti gli ingredienti contenuti).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Gusto` (la verifica va effettuata esclusivamente sul nome).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Ingrediente`, lo inserisca in coda alla lista di ingredienti.
6. Possedere il metodo `ingrediente` che, dato il nome di un ingrediente, indichi se tale ingrediente è contenuto o meno nel gusto a disposizione.

Esercizio 6 (7 punti)

Si scriva un'applicazione per Mattia De Gianni che:

1. Crei un insieme di oggetti `Gusto`.
2. Crei un oggetto `Gusto`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. all'interno dell'insieme di cui al punto 1. , controllando che tale inserimento sia possibile.
4. Crei un oggetto `Ingrediente`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno del gusto di cui al punto 2.
6. Letto da tastiera il nome di un ingrediente, stampi a video il nome dei gusti che contengono tale ingrediente all'interno dell'insieme di cui al punto 1.
7. Letto da tastiera il nome di un ingrediente, stampi a video il numero totale di barattoli dei gusti che non contengono tale ingrediente all'interno dell'insieme di cui al punto 1.

Soluzione Esercizio 2

$-44_{10} = 11010100_2$
$-85_{10} = 10101011_2$

$$\begin{array}{r} 11010100+ \\ \underline{10101011} \\ 01111111 \end{array}$$

$01111111_2 = 127_{10}$

Il numero rappresentato è diverso dall'originale (–129) poiché quest'ultimo non è rappresentabile con soli 8 bit.

Soluzione Esercizio 3

Domanda 1:

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

Domanda 2:

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

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

$$= \frac{N(N-1)}{2} + 2N + \frac{N+1}{2} = \frac{N^2}{2} + 2N + \frac{1}{2}$$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 4

```
class Ingrediente implements Comparable<Ingrediente> {
  private String nome;
  private int calorie;

  public Ingrediente(String nome, int calorie) {
    this.nome = nome;
    this.calorie = calorie;
  }

  public String getNome() { return nome; }
  public int getCalorie() { return calorie; }

  public String toString() {
    return nome + " (" + calorie + ")";
  }

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

  public int compareTo(Ingrediente i) {
    int ret = calorie - i.calorie;
    if(ret==0) ret = nome.compareTo(i.nome);
    return ret;
  }
}
```

Soluzione Esercizio 5

```
import java.util.*;

class Gusto {
  private String nome;
  private int barattoli;
  private List<Ingrediente> l;
  public Gusto(String nome, int barattoli) {
    this.nome = nome;
    this.barattoli = barattoli;
    l = new LinkedList< Ingrediente >();
  }

  public String getNome() { return nome; }
  public int getBarattoli() { return barattoli; }
  public String toString() {
    return nome+ "(" + barattoli + "): " + l;
  }

  public boolean equals(Object o) { return equals((Gusto) o); }
  public boolean equals(Gusto g){ return nome.equals(g.nome); }

  public void aggiungi(Ingrediente i) { l.add(i); }
  public boolean ingrediente(String s) {
    for(Ingrediente i: l) if(i.getNome().equals(s)) return true;
    return false;
  }
}
```

Soluzione Esercizio 6

```
import java.util.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Gusto> s = new TreeSet<Gusto>();
    Scanner scanner = new Scanner(System.in);
    Gusto g = new Gusto(scanner.nextLine(), scanner.nextInt());
    if(l.add(g)) System.out.println("Gusto già esistente!");
    Ingrediente i = new Ingrediente(scanner.nextLine(),
      scanner.nextInt());
    g.aggiungi(i);
    String nome = scanner.nextLine();
    for(Gusto x: l)
      if(x.ingrediente(nome)) System.out.println(x.getNome());
    nome = scanner.nextLine();
    int totale = 0;
    for(Gusto x: l)
      if(!x.ingrediente(nome)) totale+=x.getBarattoli();
    System.out.println(totale);
  }
}
```