FONDAMENTI DI INFORMATICA – 2014 Luglio 2 - TEORIA – modulo B (tempo 30')

Esercizio n. 1 (punti 7)
Date due liste di interi, **L1** e **L2**, ordinate e della stessa lunghezza, si consideri la seguente funzione:

```
list  p(list  L1, list L2)
{
   list L3=NULL;
   while ((L1!=NULL) && (L2!=NULL))
          { if (L1->value <= L2->value)        /* istr. dominante */
                { L3=cons_tail(L1->value, L3);
                  L1=L1->next;  }
            else
                { L3=cons_tail(L2->value, L3);
                  L2=L2->next;  }
          }
   while (L1!=NULL)
          { L3=cons_tail(L1->value, L3);
            L1=L1->next;  }
   while (L2!=NULL)
          { L3=cons_tail(L2->value, L3);
            L2=L2->next;  }

 return L3;
}
```

a) Si descriva cosa fa la funzione **p** (quale algoritmo realizza?)
b) Si discuta la complessità del codice sapendo che la lista **L1** e la lista **L2** contengono entrambe **N** elementi. Individuare la complessità come numero di confronti del test sottolineato nella funzione **p**, nel caso peggiore.

SOLUZIONE
La funzione **p** realizza il merge delle due liste date in ingresso, e produce una lista ordinata che riporta l'unione degli elementi delle due liste in ingresso. La complessità asintotica è quindi quella dell'algoritmo di merge, O(N).

Esercizio n. 3 (punti 3)
Nel codice seguente, qual è la versione del metodo `start()` che è eseguita invocando nel `main v.start()`? (quella della classe **`Vehicle`** o della classe **`Car`**? Motivare opportunamente la risposta).

```
public class July {
    public static void main(String args[]) {
        Vehicle v = new Car();
        v.start();          }
}

class Vehicle {
    public void start() {
        System.out.println("Inside start method of Vehicle");   }
}

class Car extends Vehicle {
public void start() {
        System.out.println("Inside start method of Car"); }
}
```

SOLUZIONE
La versione del metodo `start()` che è eseguita invocando `v.start()` è quella della classe **`Car`** perché `v` (dichiarato staticamente di tipo **`Vehicle`**) referenzia un'istanza della classe **`Car`** e java determina qual è la versione di codice da eseguire sulla base della natura dell'istanza (dynamic binding).