# Corso di Studi in Informatica
# ALGORITMI E LABORATORIO – prova scritta del 5 aprile 2006
# Parte 1

Cognome e Nome_____ Matr._____

**1.** (~ punti 1) Si dimostri **per induzione matematica** che un albero binario completo di altezza **k** possiede $\mathbf{2^{k+1} - 1}$ nodi e $\mathbf{2^k}$ foglie.
Nota bene: una dimostrazione non per induzione non verrà presa in considerazione.

**2**. (~ punti 4) Si consideri la semplice classe seguente, che realizza la struttura albero binario di ricerca con elementi costituiti da interi "nudi" (cioè privi di dati aggiuntivi), con i sottoalberi vuoti realizzati semplicemente come **null** e non come oggetti (e con i metodi **add**, **find**, ecc. realizzati in un modo qualunque, ad es. ricorsivo):

```
public class BSTInt {

  private class Node {
    int element;
    Node left, right;

    Node(int element, Node left, Node right) {
       ...
    }
       ...
  }

  private Node root;
  ...
}
```

In essa si definisca un metodo **void printInterval(int inf, int sup)** che scriva su consolle, in ordine crescente, tutti gli elementi dell'albero compresi nell'intervallo **[inf .. sup]**, cioè tutti gli elementi **x** tali che **inf <= x <= sup**, naturalmente definendo un corrispondente metodo nella classe *Node*. **Si richiede che l'algoritmo sia ottimale, cioè non esegua passi inutili**.

**3**. . (~ punti 3) Si consideri la seguente variante di un noto algoritmo di ordinamento per array di elementi con chiavi intere comprese fra **0** e **k-1**:

```
  public static void csort(int k, ItemWithIntKey[] a) {
    int n = a.length;
    ItemWithIntKey[] b = new ItemWithIntKey[n];
    int[] c = new int[k];
    for(int i = 0; i < n; i++) c[a[i].key()]++;
    c[0]--;
    for(int j = 1; j < k; j++) c[j] += c[j-1];
    for(int i = 0; i < n; i++) {
      b[c[a[i].key()]] = a[i];
      c[a[i].key()]--;
    }
    copia in a l'array b
  }
```

Si dica se tale variante ordina l'array-argomento e, in caso positivo, se essa gode delle stesse proprietà della versione originale illustrata nel corso oppure no. **La risposta deve essere giustificata chiaramente.**

**4**. (~ punti 4) Si costruisca uno *heap-a-minimo* contenente gli interi 10, 2, 9, 16, 8, 6, 1, 3, 12, disegnando ad ogni passo lo heap sia sotto forma di albero che come array. Lo heap deve essere costruito in due modi diversi:

1) inserendo successivamente (nell'ordine in cui sono scritti) gli elementi nello heap inizialmente vuoto;
2) costruendo lo heap dal basso in alto, attraverso una procedura *heapify* in versione per heap-a-minimo.

**5**. (~ punti 4) Si disegni la forma del generico albero AVL sbilanciato a destra (con i sottoalberi rilevanti disegnati come triangoli); si consideri in particolare il caso in cui il ribilanciamento si ottiene con una rotazione semplice, e si spieghi per mezzo di chiari disegni come tale rotazione effettua il ribilanciamento. Si disegni poi un esempio specifico di albero AVL di interi bilanciato, tale che un opportuno inserimento provochi lo sbilanciamento a destra-destra di un suo sottoalbero: si raffiguri con dei disegni tale inserimento e il successivo ribilanciamento.

**Totale punti 16**