Esame del 29 Settembre 2006
Laboratorio di Algoritmi e Strutture Dati
Anno Accademico 2005/2006 – Matricole Dispari-Pari e Pari-Dispari

Cognome e Nome: Numero di Matricola:

Docente:

Spazio riservato alla correzione

1	2	3	4	totale
/30	/8	/37	/15	/90

Non usare altri fogli, usare solo lo spazio sottostante. Fogli differenti da questo non saranno presi in considerazione per la correzione.

1. Sia **G** un grafo direzionato in cui ciascun nodo rappresenta una città e in cui ciascun arco **(u,v)** ha associato un peso che rappresenta la distanza chilometrica per andare dalla città **u** alla città **v**.
   [18 punti] Si scriva lo pseudocodice di un algoritmo che preso in input **G**, una città **s** e un intero **k** restituisce un array contenente le città che si trovano ad una distanza chilometrica minore o uguale di **k** da **s**.
   [4 punti] Si descrivano le strutture dati utilizzate.
   [8 punti] Si analizzi la complessità dell'algoritmo proposto.

2. Si scriva la funzione **void removeAllFromStack(Object elem, Stack s)** che rimuove dallo stack **s** tutte le occorrenze dell'elemento **elem**. Se lo stack è vuoto deve lanciare l'eccezione **StackEmptyException** .

3. Un MultiSet è la generalizzazione del concetto di insieme. Un elemento può comparire più volte nello stesso MultiSet. Scrivere la classe **ListMultiSet** che implementa la seguente interfaccia

```
public interface MultiSet {
// Restituisce il numero degli elementi nel multiSet
public int size();

// Restituisce il numero degli elementi distinti nel multiSet
public int distinct();

// Restituisce true se il multiSet è vuoto
  public boolean isEmpty();

// Rimpiazza this con l'unione di this e B
public MultiSet union(MultiSet B);

// Rimpiazza this con l'intersezione di this e B
// Se un elemento p compare t1 volte in this e t2 volte in B, allora l'elemento
// p comparirà min{t1,t2} volte nell'intersezione di this e B.
public MultiSet intersect(MultiSet B);

// Rimpiazza this con la differenza di this e B
// Se un elemento p compare t1 volte in this e t2 volte in B, allora l'elemento
// p comparirà max{0, t1-t2} volte nell'intersezione di this e B.
 public MultiSet subtract(MultiSet B);
}
```

Esempi: A = {7, 4, 7, 9, 9, 2, 3, 7}, B = {10, 3, 9, 7, 3, 7}.

A.size()	restituisce	8	B.size()	restituisce	6
A.distinct()	restituisce	5	B.distinct ()	restituisce	4
A.isEmpty()	restituisce	false	B.isEmpty ()	restituisce	false
A.union(B)	restituisce	{7, 4, 7, 9, 9, 2, 3, 7, 10, 3, 9, 7, 3, 7}			
B.union(A)	restituisce	{10, 3, 9, 7, 3, 7, 7, 4, 7, 9, 9, 2, 3, 7}			
A.intersect(B)	restituisce	{7, 7, 9, 3}			
B.intersect(A)	restituisce	{7, 7, 9, 3}			
A.subtract(B)	restituisce	{7, 4, 9, 2}	B.subtract(A)	restituisce	{10, 3}

4. Si supponga che un albero binario **T** rappresenti un'espressione aritmetica. Ogni nodo contiene una stringa rappresentante un numero o un operatore (+, - , /, *). Aggiungere alla classe **LinkedBinaryTree** il metodo

public String PrintExpression()

che restituisce, sottoforma di stringa opportunamente parentesizzata, l'espressione aritmetica rappresentata da **T**. Ad esempio, se **T** è il seguente albero, allora **T**.**PrintExpression()** restituisce la stringa "(2 × (a - 1) + (3 × b))".