Cognome e Nome _____

Matricola _____

Programmazione
A.A. 2016/2017

Appello del 22 Dicembre 2016

Compito n° 1

Prima parte

Esercizio 1 (10 punti)

Scrivere un metodo

`public static int contaOccorrenze (int[] a, int n)`

che, preso come parametro un array a di numeri interi e un intero n, senza modificare il contenuto dell'array, restituisce il numero di occorrenze dell'intero n all'interno dell'array a. Se **a** vale *null*, viene restituito *0*.
Ad esempio, se a= [10, 6, 5, 4, 5, 6] e n=5 deve essere restituito 2.

Esercizio 2 (10 punti)

Scrivere un metodo

`public static boolean primaOccorrenza (int[]a, int pos)`

che, preso come parametro un array di numeri interi **a** e un intero **pos**, senza modificare il contenuto dell'array, restituisce true se e solo se l'elemento in posizione **pos** non compare nell'array **a** nelle posizioni da 0 a **pos**-1.
Se **a** vale *null*, viene restituito *false*. Se **pos** è fuori dagli indici ammissibili per l'array, viene restituito *false*.
Ad esempio, se a={5,3,7,7,3,10,5} e **pos** = 2, il metodo deve restituire *true* in quanto l'elemento 7 compare per la prima volta in posizione 2.

Esercizio 3 (13 punti)

Scrivere un metodo

`public static int[] multiIntersezione (int[]a, int[] b)`

che, presi come parametri due array di numeri interi **a** e **b**, senza modificare il contenuto degli array, crea e restituisce un nuovo array contenente l'intersezione dei multinsiemi rappresentati dagli array **a** e **b**.
Nell'intersezione di due multinsiemi **a** e **b** compaiono tutti e soli gli elementi comuni ai multinsiemi con un numero di ripetizioni uguale al minimo tra il numero di ripetizioni dell'elemento in **a** e quello in **b**.
Se **a** o **b** valgono *null*, viene restituito *null*.
Ad esempio, se a={5, 3, 7, 7, 3, 10, 5} e b={20, 5, 7, 5, 30}, il metodo deve restituire {5, 5, 7}.
Il metodo **deve** richiamare i metodi contaOccorrenze e primaOccorrenza e potrà applicare il seguente algoritmo:

- prima bisogna contare quanti elementi vanno inseriti nell'array da restituire:
  - per ogni posizione **i** dell'array **a**, se **i** è la prima occorrenza dell'elemento **a[i]**, devono essere contati un numero di elementi in numero uguale al minimo tra le occorrenze di **a[i]** in **a** e le occorrenze di **a[i]** in **b**
- bisogna creare l'array da restituire dell'opportuna lunghezza
- bisogna infine riempire l'array creato:
  - per ogni posizione **i** dell'array **a**, se **i** è la prima occorrenza dell'elemento **a[i]**, devono essere inseriti un numero di elementi uguali ad **a[i]** in numero uguale al minimo tra le occorrenze di **a[i]** in **a** e le occorrenze di **a[i]** in **b**

Seconda parte

Esercizio 4 (10 punti)

Cosa stampa il seguente programma Java?

```
class A {
      private double x;

      public A (double x){
            this.x=x;
      }
      public double getX(){
            return x;
      }
      double metodo (A a){
            return getX()-3*a.getX();
      }
}


class B extends A{
      public B(double x){
            super (2*x);
      }
      double metodo (A a){
            return this.getX()+
                  5*a.getX();
      }
}
```

```
public class MainClass {
      public static void main
                  (String[] args) {
       A a = new A(3);
       A b = new B(2);
       double y = a.metodo(b);
       double z = b.metodo(a);
       System.out.println(a.getX());
       System.out.println(b.getX());
       System.out.println(y);
       System.out.println(z);
      }
}
```

Esercizio 5 (23 punti)

Si progetti una classe **MultiElemento** con le variabili di istanza **valore** (tipo double, final) e **ripetizioni** (tipo int). La classe deve avere:

- un costruttore che crea un MultiElemento dati valore e ripetizioni (se ripetizioni è minore di 1, viene impostato ad 1);
- un costruttore che crea un MultiElemento dato il valore (impostando ripetizioni ad 1);
- un metodo getValore();
- un metodo getRipetizioni();
- un metodo setRipetizioni (int ripetizioni) che imposta il campo ripetizioni al valore dato come parametro (se ripetizioni è minore di 1, viene impostato ad 1);
- un metodo toString() che restituisce la descrizione nel seguente modo (nell'esempio seguente il campo valore è 3.14 e il campo ripetizioni è 5):
  `elemento 3.14 occorrenze 5`

Si progetti una classe **MultiInsieme** con una variabile di istanza **set** di tipo ArrayList<MultiElemento> che rappresenti un multinsieme. La classe deve avere:

- un costruttore che crea un MultiInsieme vuoto
- un metodo getRipetizioni (double valore) che restituisce il numero di ripetizioni relative all'elemento valore contenuto nell'insieme
- un metodo addElemento (double valore, int ripetizioni) che aggiunge al multiInsieme l'elemento valore con il numero di ripetizioni indicato. ***Attenzione***: se nel multinsieme c'è già un elemento con lo stesso valore, viene solo aggiornato il suo campo ripetizioni, altrimenti viene aggiunto all'arraylist un nuovo MultiElemento.

Cognome e Nome _____

Matricola _____

Programmazione
A.A. 2016/2017

Appello del 22 Dicembre 2016

Compito n° 2

Prima parte

Esercizio 1 (10 punti)

Scrivere un metodo

`public static int contaOccorrenze (int[] a, int n)`

che, preso come parametro un array a di numeri interi e un intero n, senza modificare il contenuto dell'array, restituisce il numero di occorrenze dell'intero n all'interno dell'array a. Se **a** vale *null*, viene restituito *0*.
Ad esempio, se a= [10, 6, 5, 4, 5, 6] e n=5 deve essere restituito 2.

Esercizio 2 (10 punti)

Scrivere un metodo

`public static boolean ultimaOccorrenza (int[]a, int pos)`

che, preso come parametro un array di numeri interi **a** e un intero **pos**, senza modificare il contenuto dell'array, restituisce true se e solo se l'elemento in posizione **pos** non compare nell'array **a** nelle posizioni da **pos**+1 in poi.
Se **a** vale *null*, viene restituito *false*. Se **pos** è fuori dagli indici ammissibili per l'array, viene restituito *false*.
Ad esempio, se a={5,3,7,7,3,10,5} e **pos** = 3, il metodo deve restituire *true* in quanto l'elemento 7 compare per l'ultima volta in posizione 3.

Esercizio 3 (13 punti)

Scrivere un metodo

`public static int[] multiIntersezione (int[]a, int[] b)`

che, presi come parametri due array di numeri interi **a** e **b**, senza modificare il contenuto degli array, crea e restituisce un nuovo array contenente l'intersezione dei multinsiemi rappresentati dagli array **a** e **b**.
Nell'intersezione di due multinsiemi **a** e **b** compaiono tutti e soli gli elementi comuni ai multinsiemi con un numero di ripetizioni uguale al minimo tra il numero di ripetizioni dell'elemento in **a** e quello in **b**.
Se **a** o **b** valgono *null*, viene restituito *null*.
Ad esempio, se a={5, 3, 7, 7, 3, 10, 5} e b={20, 5, 7, 5, 30}, il metodo deve restituire {5, 5, 7}.
Il metodo **deve** richiamare i metodi contaOccorrenze e ultimaOccorrenza e potrà applicare il seguente algoritmo:

- prima bisogna contare quanti elementi vanno inseriti nell'array da restituire:
  - per ogni posizione **i** dell'array **a**, se **i** è l'ultima occorrenza dell'elemento **a[i]**, devono essere contati un numero di elementi in numero uguale al minimo tra le occorrenze di **a[i]** in **a** e le occorrenze di **a[i]** in **b**
- bisogna creare l'array da restituire dell'opportuna lunghezza
- bisogna infine riempire l'array creato:
  - per ogni posizione **i** dell'array **a**, se **i** è l'ultima occorrenza dell'elemento **a[i]**, devono essere inseriti un numero di elementi uguali ad **a[i]** in numero uguale al minimo tra le occorrenze di **a[i]** in **a** e le occorrenze di **a[i]** in **b**

Seconda parte

Esercizio 4 (10 punti)

Cosa stampa il seguente programma Java?

```
class A {
      private double x;

      public A (double x){
            this.x=x;
      }
      public double getX(){
            return x;
      }
      double metodo (A a){
            return getX()-2*a.getX();
      }
}


class B extends A{
      public B(double x){
            super (3*x);
      }
      double metodo (A a){
            return this.getX()+
                  7*a.getX();
      }
}
```

```
public class MainClass {
      public static void main
                  (String[] args) {
       A a = new A(5);
       A b = new B(2);
       double y = a.metodo(b);
       double z = b.metodo(a);
       System.out.println(a.getX());
       System.out.println(b.getX());
       System.out.println(y);
       System.out.println(z);
      }
}
```

Esercizio 5 (23 punti)

Si progetti una classe **MultiElemento** con le variabili di istanza **valore** (tipo double, final) e **ripetizioni** (tipo int). La classe deve avere:

- un costruttore che crea un MultiElemento dati valore e ripetizioni (se ripetizioni è minore di 1, viene impostato ad 1);
- un costruttore che crea un MultiElemento dato il valore (impostando ripetizioni ad 1);
- un metodo getValore();
- un metodo getRipetizioni();
- un metodo setRipetizioni (int ripetizioni) che imposta il campo ripetizioni al valore dato come parametro (se ripetizioni è minore di 1, viene impostato ad 1);
- un metodo toString() che restituisce la descrizione nel seguente modo (nell'esempio seguente il campo valore è 3.14 e il campo ripetizioni è 5):
  `elemento 3.14 occorrenze 5`

Si progetti una classe **MultiInsieme** con una variabile di istanza **set** di tipo ArrayList<MultiElemento> che rappresenti un multiinsieme. La classe deve avere:

- un costruttore che crea un MultiInsieme vuoto
- un metodo getRipetizioni (double valore) che restituisce il numero di ripetizioni relative all'elemento valore contenuto nell'insieme
- un metodo addElemento (double valore, int ripetizioni) che aggiunge al multiInsieme l'elemento valore con il numero di ripetizioni indicato. Attenzione: se nel multinsieme c'è già un elemento con lo stesso valore, viene solo aggiornato il suo campo ripetizioni, altrimenti viene aggiunto all'arraylist un nuovo MultiElemento.