Laboratorio di Programmazione
(Corso di Laurea in Informatica)

APPELLO del 22 Gennaio 2018

Avvertenze

- I programmi realizzati DEVONO rispettare le specifiche funzionali fornite (input/output).
- Nello svolgimento dell'elaborato è possibile usare qualunque classe delle librerie standard di Java.
- Si raccomanda di salvare, compilare e fare upload delle soluzioni con una certa regolarità. Il sistema di upload mantiene tutte le versioni di OGNI file, la correzione verrà effettuata consultando l'ultima versione dei file consegnati.
- Avete a disposizione un file `_CheckNomiClassiMetodi.java` che serve per fare un controllo sui nomi: se questa classe compila senza errori vuol dire che i NOMI di classi e metodi del vostro elaborato sono corretti, null'altro!
- Per la procedura di **consegna** si veda in fondo al documento.

---

1 Statistiche

1.1 Descrizione

In una classe "Statistiche" (rispettare il nome della classe!) definire i due metodi **statici** e **pubblici** descritti di seguito:

- `int[] frequenze (int data[])`
  assumendo che l'array `data` abbia lunghezza > 0 e che i valori memorizzati in esso siano ≥ 0, restituisce un (altro) array con le frequenze dei valori
- `double mediana (int data[])`
  assumendo che l'array `data` abbia lunghezza > 0 calcola e restituisce la mediana dei valori presenti nell'array.

Si ricorda che la mediana di una serie di valori è:

- il valore che occupa la posizione centrale dei dati ordinati (in modo crescente o decrescente), se i dati sono in numero dispari;
- la media dei due valori che occupano le due posizioni centrali dei dati ordinati, se i dati sono in numero pari.

Di seguito viene fornito un `main` che può essere utilizzato per un test *preliminare* e modificato a piacere. Si ricorda che la firma dei metodi NON può essere cambiata in alcun modo.

```
/*esegue un semplice test di frequenze e mediana*/
public static void main(String[] args) {
        int[] data = {1,1,3,2,4,5,1,4,6,7,3,5,5,8,9,10,0,0,2,1};
        int[] frequenza = frequenze(data);
        System.out.println("serie di dati:");
        for (int x : data)
            System.out.print(x+" ");
        System.out.println("\nfrequenze:");
        for (int i = 0; i < frequenza.length ; ++i)
            System.out.printf("%2d: %d\n", i, frequenza[i]);
        System.out.println("mediana: "+ mediana(data)+'\n'); // dovrebbe stampare 3.5
    }
```

Nota bene: il programma verrà testato tramite una classe esterna.

---

2 Parcheggio

2.1 Descrizione

Un supermercato dispone di un parcheggio con una serie di posti aventi ciascuno una dimensione espressa da un numero intero compreso fra 1 e 10 (estremi inclusi). Quando un veicolo di una certa dimensione arriva al supermercato, parcheggia nel primo posto libero esattamente della sua dimensione; se non esiste alcun posto libero del genere, il veicolo non può parcheggiare.
Nel seguito identifichiamo sia un posto del parcheggio sia una vettura con l'intero che ne denota la dimensione.
Scrivere un programma (una classe "Parcheggio" dotata del metodo `main`) che legge da standard input due righe di interi (che si assume siano nell'intervallo $1 \dots 10$)
$p_1\ p_2\ \dots\ p_n$
$v_1\ v_2\ \dots\ v_m$
con $1 \leq n \leq 20$ e $m \geq 1$, dove la prima riga è l'elenco dei posti liberi, la seconda riga l'elenco delle vetture in arrivo al supermercato. Il programma deve emettere sul flusso di output l'elenco dei posti rimasti liberi dopo che tutte le vetture che potevano farlo hanno parcheggiato.

Ad esempio, supponiamo che le righe di ingresso siano
2 5 3 5 1 4 5
7 5 4 5 2 4
La prima vettura (dimensione 7) non può parcheggiare, in quanto non esiste alcun posto libero di dimensione 7.
La seconda vettura (dimensione 5) può parcheggiare in quanto ci sono tre posti liberi di dimensione 5 e occuperà il primo di essi (il secondo posto del parcheggio).
La terza vettura (dimensione 4) parcheggia nell'unico posto libero di dimensione 4.
La quarta vettura (dimensione 5) occupa il primo posto libero di dimensione 5 (il quarto posto del parcheggio).
E cosí di seguito. Alla fine, l'elenco dei posti liberi è
3 1 5

2.2 Esempio

Eseguendo il programma
`java Parcheggio`
avendo sul flusso di input
```
6 2 3 9 7 3 5 5 7 8 7 3 6
7 5 3 1 3 3 6 3 2 7 4 8 5 2 4
```
viene emesso il seguente testo
```
9 7 6
```

---

3 Coefficienti binomiali

3.1 Descrizione

Il coefficiente binomiale $\binom{n}{k}$ è il numero di modi di scegliere un sottoinsieme di k elementi da un insieme di n elementi. Per calcolarlo si può utilizzare la *regola di Pascal*, che esprime $\binom{n}{k}$ in termini di coefficienti binomiali piú piccoli:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Per $k = 0$, il coefficiente binomiale è 1; altrimenti per $n = 0$, il coefficiente binomiale è 0.

Scrivere un programma (una classe "CoefficienteBinomiale") dotata di un metodo ricorsivo che calcola $\binom{n}{k}$ per n e k numeri naturali utilizzando la regola di Pascal, e di un metodo `main` che, dati due numeri naturali n e k (in quest'ordine) passati da linea di comando, emetta su standard output il valore di $\binom{n}{k}$.

Esempi

```
$java CoefficienteBinomiale 6 2
15
$java CoefficienteBinomiale 5 0
1
$java CoefficienteBinomiale 0 3
0
$java CoefficienteBinomiale 14 14
1
$java CoefficienteBinomiale 0 0
1
```

Consegna

Si ricorda che le classi devono essere tutte *public* e che vanno consegnati tutti (e soli) i file *.java* prodotti. NON vanno consegnati i *.class*. Per la consegna, eseguite l'upload dei SINGOLI file

sorgente (NON un file archivio!) dalla pagina web: http://upload.di.unimi.it nella sessione che vi è stata indicata.

*** ATTENZIONE!!! ***

NON VERRANNO VALUTATI GLI ELABORATI CON ERRORI DI COMPILAZIONE O LE CONSEGNE CHE NON RISPETTANO LE SPECIFICHE (ad esempio consegnare un archivio zippato è sbagliato)
UN SINGOLO ERRORE DI COMPILAZIONE O DI PROCEDURA INVALIDA **TUTTO** L'ELABORATO.

Per ritirarsi fare l'upload di un file vuoto di nome `ritirato.txt`.
