Corso di Programmazione 1

Quarta Esercitazione di Laboratorio

Esercizio 1

Scrivere un programma Java che faccia inserire all'utente un numero intero positivo *n*, e successivamente una sequenza di numeri interi positivi. La sequenza non termina finché non viene reinserito il numero *n*.

Esercizio 2

Scrivere un programma Java che chieda all'utente di inserire da tastiera un numero intero *n* non negativo (cioè positivo o nullo), e calcoli (usando un opportuno ciclo `while`) il *fattoriale* di *n*, cioè il prodotto di tutti i numeri compresi tra 1 e *n*. Si ricorda che, per definizione, il fattoriale di 0 è 1.

Esercizio 3

Scrivere un programma Java che legga da tastiera due numeri interi, *base* e *esponente*, il secondo dei quali deve essere maggiore o uguale a 0, e calcoli l'elevamento del primo numero alla potenza indicata dal secondo (**non** utilizzando il metodo Math.pow()). Si noti che *base* può anche essere negativo o nullo.

Esercizio 4

Scrivere un programma Java che legga da tastiera un intero positivo, rappresentante la capacità in kg di uno zaino, e riceva una sequenza di interi positivi rappresentanti i pesi degli oggetti da inserirvi, fino a che la somma dei pesi non eccede la capacità oppure viene letto da tastiera uno 0.
Al termine il programma deve stampare a video la capacità dello zaino, il numero e il peso totale degli oggetti in esso contenuti, il peso dell'oggetto più pesante, il peso dell'oggetto più leggero, e il peso medio degli oggetti presenti nello zaino.

Esercizio 5

Scrivere un programma Java che, letti da tastiera due numeri interi positivi *n* ed *m*, calcoli il *quoziente* e il *resto* della divisione intera tra *n* ed *m* usando solo **sottrazioni successive** (senza usare, quindi, gli operatori / e %).

Esercizio 6

Scrivere un programma Java che, lette da tastiera due stringhe formate da 0 e 1 di uguale lunghezza, consideri tali stringhe come rappresentanti gli elementi di due insiemi. Ad esempio, se le stringhe inserite dall'utente sono 01101 e 10110, abbiamo che:

- il primo insieme, rappresentato dalla stringa 01101, contiene il secondo, terzo e quinto elemento;
- il secondo insieme, rappresentato dalla stringa 10110, contiene il primo, terzo e quarto elemento.

Dopo aver verificato la validità delle stringhe inserite (devono essere composte solamente da 0 e 1, e devono avere la stessa lunghezza), stampare a video le due stringhe corrispondenti all'*unione* e all'*intersezione* dei due insiemi. Quindi, se l'utente inserisce le due stringhe 01101 e 10110 indicate sopra, il programma stamperà la stringa 11111 come rappresentante dell'unione, e 00100 come rappresentante dell'intersezione.

Esercizio 7

Scrivere un programma Java che, letto da tastiera un numero *n* maggiore o uguale a 0, calcoli (usando un opportuno ciclo `while`) l'*n*-esimo numero della sequenza 0, 1, 1, 2, 3, 5, 8, 13, 21, … di Fibonacci. Si ricorda che l'*n*-esimo numero di Fibonacci può essere definito come segue:

$fib(0) = 0$

$fib(1) = 1$

$fib(n) = fib(n-1) + fib(n-2)$ per $n > 1$

Tuttavia, questa è una definizione *ricorsiva*, mentre noi vogliamo un metodo *iterativo* per calcolare il valore di $fib(n)$.

Esercizio 8

Scrivere un programma Java che, letti da tastiera due numeri interi positivi *n* ed *m*, calcoli il *massimo comun divisore* MCD(*n*,*m*) usando l'algoritmo di Euclide, che si basa sulle seguenti proprietà:

- $MCD(x,x) = x$
- $MCD(x,y) = MCD(y,x)$
- $MCD(x,y) = MCD(x-y,y)$ se $x > y$