Programmazione e Algoritmi 1

A.A. 2023/24 — Compito del 3 giugno 2024 – Versione Java

prof. Gianluca Amato

Gli esercizi di programmazione saranno valutati sulla base della correttezza, efficienza e comprensibilità della soluzione proposta. In generale, **se volete usare una funzione o un metodo che non è stato presentato a lezione, chiedete prima al docente se è consentito**.

Esercizio 1 (5 punti)

Spiegare la differenza esistente tra linguaggi a basso e ad alto livello, e tra interpreti e compilatori.

Esercizio 2 (8 punti)

Si consideri il seguente codice Java, e se ne scriva la traccia di esecuzione negli appositi moduli.

```
1  public class Mistero {
2      public static String mistero(String l, int n, String r) {
3          if (n == l.length())
4              return r;
5          else if (n % 2 == 0)
6              return mistero(l, n + 1, l.charAt(n) + r);
7          else
8              return mistero(l, n + 1, r);
9      }
10
11     public static void main(String[] args) {
12         String x = mistero("ciao", 0, "");
13         System.out.println(x);
14     }
15 }
```

Si ricordi che `l.charAt(n)` restituisce il carattere in posizione n della stringa l.

Esercizio 3 (5 punti)

Scrivere il metodo statico `sommaOpposti` che prende come parametro un array di numeri interi e restituisce `true` se le somme del primo e ultimo elemento dell'array, del secondo e del penultimo, del terzo e del terzultimo, e così, via sono tutte uguali. In caso contrario il metodo restituisce `false`. Ecco alcuni esempi:

- `sommaOpposti(new int[] {1, 2, 3, 4, 5, 6, 7, 8, 9, 10})` restituisce `true` perché $1+10 = 2+9 = 3+8 = 4+7 = 5+6 = 11$;

- `sommaOpposti(new int[] {1, 1, 1})` restituisce `true` perché il primo e ultimo elemento sono due numeri uno, che sommati fanno 2; analogamente, anche il secondo e il penultimo elemento sono due numeri uno (sebbene siano lo stesso elemento della lista) e sommati fanno anche loro 2;
- `sommaOpposti(new int[] {1, 2, 2, 10})` restituisce `false` perché $1 + 10 \neq 2 + 2$.
- `sommaOpposti(new int[0])` restituisce `true` perché non ci sono coppie di elementi da sommare.

Esercizio 4 (5 punti)

Scrivere alcuni test nel framework `JUnit` per verificare il corretto funzionamento di `sommaOpposti`. In particolare, si deve controllare che il valore di ritorno sia corretto per:

- tutti gli array di esempio dell'Esercizio 1;
- dieci array generati casualmente in cui tutti gli elementi sono uguali: devono essere generati casualmente sia l'elemento da inserire nell'array sia la lunghezza dell'array (entrambi numeri interi nell'intervallo da 0 a 99).

Gi studenti degli a.a. 2021/22 e precedenti, invece di usare il framework `JUnit`, possono scrivere un normale programma Java che esegue i test e ne visualizza i risultati (passato / non passato) sullo schermo.

Esercizio 5 (5 punti)

Scrivere un metodo statico `mascheraMatrice` che prende come parametro un array bidimnsionale di interi `m` e un valore intero `v`. Il metodo deve restituire un nuovo array bidimensionale che ha le stesse dimensioni di `m` e che contiene `true` nelle posizioni in cui il valore corrispondente in `m` è strettamente maggiore di `v`, e `false` altrimenti. L'array `m` non deve essere modificato.

Esercizio 6 (5 punti)

Scrivere un programma che prende in input da terminale due numeri interi `n` ed `m` e disegna una griglia formata da $n \times m$ rettangoli, ognuno formato da 5 spazi (vedere esempio sotto). Al centro di ogni rettangolo deve essere scritto un numero progressivo che parte da 1 e si incrementa di 1 ad ogni rettangolo, secondo l'ordine consueto di lettura. Quello che segue è un esempio di interazione con il programma. In rosso trovate quello scritto dall'utente, in nero l'output del programma. I bordi della griglia (realizzati con il carattere meno e la barra verticale) sono obbligatori.

```
Inserisci n (numero righe): 3
Inserisci m (numero colonne): 4
-------------------------
|  1  |  2  |  3  |  4  |
-------------------------
|  5  |  6  |  7  |  8  |
-------------------------
|  9  | 10  | 11  | 12  |
-------------------------
```