Programmazione e Algoritmi 1

A.A. 2023/24 — Compito del 3 giugno 2024

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 Python, e se ne scriva la traccia di esecuzione negli appositi moduli.

```
1  def mistero(l, n, r):
2    if n == len(l):
3      return r
4    elif n % 2 == 0:
5      return mistero(l, n+1, l[n] + r)
6    else:
7      return mistero(l, n+1, r)
8
9  x = mistero("ciao", 0, "")
10 print(x)
```

Esercizio 3 (5 punti)

Scrivere la funzione `somma_opposti` che prende come parametro una lista di numeri interi e restituisce `True` se le somme del primo e ultimo elemento, del secondo e del penultimo, del terzo e del terzultimo, e così, via sono tutte uguali. In caso contrario la funzione restituisce `False`. Ecco alcuni esempi:

- `somma_opposti([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])` restituisce `True` perché $1 + 10 = 2 + 9 = 3 + 8 = 4 + 7 = 5 + 6 = 11$;
- `somma_opposti([1, 1, 1])` restituisce `True` perché il primo e ultimo elementi sono due numeri uno, che sommati fanno 2; analogamente, anche il secondo e il penultimo elemento sono due numeri uno (sebbene sia lo stesso elemento della lista) e sommati fanno anche loro 2;
- `somma_opposti([1, 2, 2, 10])` restituisce `False` perché $1 + 10 \neq 2 + 2$.
- `somma_opposti([])` restituisce `True` perché non ci sono coppie di elementi da sommare.

Esercizio 4 (5 punti)

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

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

Esercizio 5 (5 punti)

Scrivere una funzione `maschera_matrice` che prende come parametro una matrice `m`, implementata come lista di liste, ed un valore `v`. La funzione deve restituire una nuova matrice 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. La matrice `m` non deve essere modificata.

Esercizio 6 (5 punti)

Scrivere una programma che prende in input da terminale due numeri interi `n` ed `m`. Utilizzando la libreria `ezgraphics`, il programma disegna una griglia formata da $n \times m$ quadrati di lato 40 pixel. Al centro di ogni quadrato deve essere scritto un numero progressivo che parte da 1 e si incrementa di 1 ad ogni quadrato, secondo l'ordine consueto di lettura.

Ad esempio, se l'utente immette i numeri 3 e 4 per, rispettivamente, il numero di righe e di colonne, il programma dovrà visualizzare la seguente griglia:

1	2	3	4
5	6	7	8
9	10	11	12