Programmazione e Algoritmi 1

A.A. 2023/24 — Compito dell'11 settembre 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)

Descrivere gli algoritmi di ricerca lineare e di ricerca binaria. Per ognuno di essi:

- specificare cosa si aspetta in input l'algoritmo;
- specificare cosa produce come risultato l'algoritmo;
- illustrarne a grandi linee il funzionamento;
- indicarne la complessità computazionale.

Determinare inoltre due input specifici, validi per entrambi gli algoritmi: uno in cui è più efficiente (fa meno confronti) la ricerca lineare, ed uno in cui è più efficiente la ricerca binaria (e spiegare brevemente il motivo).

Esercizio 2 (8 punti)

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

```
1  def f1(n):
2      if n == 0:
3          return True
4      else:
5          return f2(n - 1)
6
7  def f2(n):
8      if n == 0:
9          return False
10     else:
11         return f1(n - 1)
12
13 print(f1(4))
```

Esercizio 3 (5 punti)

Scrivere una funzione `squares(n)` che restituisce la lista di tutti i quadrati perfetti minori o uguali ad n. Si può assumere che n sia un intero maggiore o uguale a 0. Ad esempio, `squares(30)` restituisce `[0, 1, 4, 9, 16, 25]`. Si ricorda che un quadrato perfetto è un numero del tipo x^2 per qualche intero x, ed infatti 0, 1, 4, 9, 16 e 25 sono rispettivamente 0^2, 1^2, 2^2, 3^2, 4^2 e 5^2.

Esercizio 4 (5 punti)

Scrivere due test nel framework `pytest` per verificare il corretto funzionamento di `squares`. In particolare, si implementino due test.

- un test che verifichi se le seguenti chiamate di funzione restituiscono il risultato desiderato:
  - `squares(5)`
  - `squares(1)`
- un test per il caso in cui il numero n è un intero positivo o nullo generato casualmente. In questo caso, il test deve controllare solo che la lista restituita dalla funzione non sia vuota e che l'ultimo elemento sia minore o uguale ad n.

Esercizio 5 (5 punti)

Si utilizzi la funzione `squares` per scrivere un programma che legge un numero intero n da tastiera e stampa la radice quadrata di n, approsimata all'intero più piccolo (ad esempio, se l'input immesso da tastiera è 30 il risultato sarà 5). In aggiunta, se il numero immesso è negativo, il programma deve stampare un messaggio di errore e chiedere di reinserire il numero. Questo è un esempio di interazione: in rosso quello scritto dall'utente, in nero l'output del programma.

```
Immetti un numero positivo o nullo: -5
Errore: il numero deve essere positivo o nullo.
Immetti un numero positivo o nullo: 30
La radice quadrata intera di 30 è 5.
```

Esercizio 6 (5 punti)

Scrivere un programma che legge il file `disegno.txt`. Ogni riga del file è composto da due numeri interi separati da spazio, che sono da interpretare come le coordinate x ed y di un punto. Il programma, utilizzando la libreria `ezgraphics`, disegna un poligono con i punti letti dal file. Ad esempio, se il file `disegno.txt` contiene:

```
100 100
150 100
200 50
250 100
300 100
```

il progrmma traccia una linea dal punto di coordinate $(100, 100)$ al punto $(150, 100)$, poi una linea dal punto $(150, 100)$ al punto $(200, 50)$, poi una linea dal punto $(200, 50)$ al punto $(250, 100)$ e così via. In particolare, con l'esempio di sopra, l'output grafico dovrebbe essere questo: