

Programmazione e Algoritmi 1 - Prova pratica

A.A. 2024/25 — Compito del 8/10/2025

prof. Gianluca Amato

Esercizio 1 (8 punti)

Scrivere una programma che legge un file di testo dal nome `input.txt`. Ogni riga del file è composta da due numeri interi separati da uno spazio, da interpretare come coordinate x ed y di un pixel in una finestra grafica. Tramite la libreria `ezgraphics`, per ogni riga del file disegnare un cerchio di raggio 10 pixel centrato alle coordinate indicate in quella riga. Il cerchio deve essere riempito alternativamente con i colori rosso e verde.

Ad esempio, se il file `input.txt` contiene i valori

```
50 70
90 200
10 10
```

il programma disegnerà tre cerchi: il primo rosso centrato alle coordinate (50,70), il secondo verde centrato alle coordinate (90,200) e il terzo di nuovo rosso alle coordinate (10,10).

Esercizio 2 (6 punti)

Scrivere la funzione `cumsum` (abbreviazione di *cumulative sum*) che prende come parametro una lista l e restituisce una nuova lista della stessa lunghezza in cui ogni elemento è la somma degli elementi di l fino a quell'indice. Ad esempio, il risultato di `cumsum([1, 2, 3, 4])` è `[1, 3, 6, 10]` perché $1 + 2 = 3$, $1 + 2 + 3 = 6$ e $1 + 2 + 3 + 4 = 10$.

Scrivere quindi la funzione `cumsum_inplace` che è simile a `cumsum`, ma opera sul posto: non restituisce nulla, e invece modifica direttamente la lista l passata come argomento. Ad esempio, se `l=[1, 2, 3, 4]` allora dopo l'esecuzione di `cumsum_inplace(l)` la lista l diventa `[1, 3, 6, 10]`.

Esercizio 3 (3 punti)

Utilizzare il framework `pytest` per collaudare le funzioni `cumsum` e `cumsum_inplace` dell'esercizio precedente. Si considerino in particolare, per entrambe le funzioni, i casi corrispondenti ai seguenti parametri in input:

- l'esempio fornito nel testo dell'esercizio;
- una lista con un solo elemento;
- una lista vuota.

Programmazione e Algoritmi 1 - Prova scritta

A.A. 2024/25 — Compito del 8/10/2025

prof. Gianluca Amato

Esercizio 4 (8 punti)

Eseguire passo-passo il seguente codice Python, utilizzando l'apposito modulo:

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

Esercizio 5 (4 punti)

Descrivere l'algoritmo di ordinamento per fusione (merge sort): input, output, funzionamento, vantaggi e svantaggi rispetto ad altri algoritmi che risolvono lo stesso problema. Successivamente, illustrarne il funzionamento in dettaglio applicandolo alla lista ["quel", "ramo", "del", "lago", "di", "come", "che", "volge", "a", "mezzogiorno"].

Esercizio 6 (5 punti)

Scrivere la funzione `alternate_strings` che prende come parametri due stringhe e restituisce la stringa ottenuta alternando i caratteri delle due stringhe. Quando la stringa più corta termina, i restanti caratteri vengono presi dalla stringa più lunga. Ad esempio, `alternate_strings("abc", "12345")` restituisce "a1b2c345".