Laboratorio 3

Elementi di Informatica e Programmazione

Esercizio 1: Rimozione doppie

Scrivere il programma `stripDuplicates.py` che legga un'unica riga di testo e, poi, la visualizzi priva di eventuali caratteri consecutivi duplicati Esempio:

- legge `Pippo` e scrive `Pipo`
- legge `pseudo---casuale` e scrive `pseudo-casuale`
- legge `mammmma` e scrive `mama`

Esercizio 2: Tavole Pitagoriche delle potenze

Scrivere il programma `tableOfPowers.py` che visualizzi una tavola pitagorica dopo aver chiesto all'utente i valori massimi della base e dell'esponente, che devono essere numeri interi positivi. Ogni riga della tabella deve contenere le potenze consecutive di una stessa base, con base crescente dall'alto in basso (come in una normale tavola pitagorica). Porre particolare cura nell'impaginazione della tabella, che deve rispettare queste regole:

- In ciascuna riga, due valori consecutivi devono essere separati da almeno uno spazio.
- In ciascuna colonna, tutti i valori devono essere "allineati a destra", cioè la cifra che rappresenta le unità di un valore deve trovarsi nella stessa colonna della cifra che rappresenta le unità degli altri valori.
- La lunghezza delle righe deve essere quella minima che risulta compatibile con le regole precedenti.

Ad esempio, quando il valore massimo della base è 10 e il massimo esponente è 5, la tabella formattata correttamente è la seguente.

```
 1   1    1     1      1
 2   4    8    16     32
 3   9   27    81    243
 4  16   64   256   1024
```

```
 5  25  125   625   3125
 6  36  216  1296   7776
 7  49  343  2401  16807
 8  64  512  4096  32768
 9  81  729  6561  59049
10 100 1000 10000 100000
```

Suggerimenti:

- potete usare gli operatori di formattazione delle stringhe. Ricordate che `"%10d"` formatta un numero intero in modo che occupi *almeno* 10 spazi. Se ha meno di 10 cifre allora un numero appropriato di spazi verrà aggiunto *all'inizio*.
- notate che ogni colonna ha una larghezza diversa. Qual è la larghezza di ogni colonna? Usate la risposta a questa domanda per costruire la stringa di formattazione di ogni elemento della tabella.

Esercizio 3: quadrati magici

Scrivere il programma `isPerfectMagicSquare.py` che verifichi se un quadrato di numeri interi è un "quadrato magico perfetto". Una disposizione bidimensionale di numeri tutti diversi avente dimensione $n \times n$ è un quadrato magico se la somma degli elementi di ogni riga, di ogni colonna e delle due diagonali principali ha lo stesso valore, detto "costante magica" o "somma magica" del quadrato. Se, in aggiunta alle condizioni precedenti, i numeri presenti nel quadrato di dimensione n sono i numeri interi da 1 a n^2, allora il quadrato magico si dice "perfetto". Il quadrato magico di dimensione 1 può essere considerato un caso limite o degenere, ma il programma deve riconoscerlo come corretto (non esistono, invece, quadrati magici di dimensione 2, né perfetti né imperfetti, come è facile dimostrare matematicamente).

L'utente introduce i numeri del (presunto) quadrato perfetto riga per riga, in sequenza, separati da almeno uno spazio (il numero di righe viene dedotto dal programma esaminando il numero di colonne presenti nella prima riga). A quel punto, il programma deve intraprendere le azioni seguenti:

1. verificare che nei dati forniti in input ci siano tante righe quante colonne e che tutte le righe/colonne abbiano la stessa lunghezza: in caso contrario, il programma termina segnalando un fallimento;
2. verificare che la sequenza di valori introdotta contenga tutti (e soli) i numeri da 1 a n2, senza ripetizioni: in caso contrario, il programma termina segnalando un fallimento;
3. visualizzare la matrice, incolonnata correttamente
4. verificare la validità delle regole del quadrato magico, interrompendo la verifica con la segnalazione di fallimento non appena una regola non sia rispettata;
5. segnalare il successo della verifica. Collaudare il programma con gli esempi riportati sopra riportati.

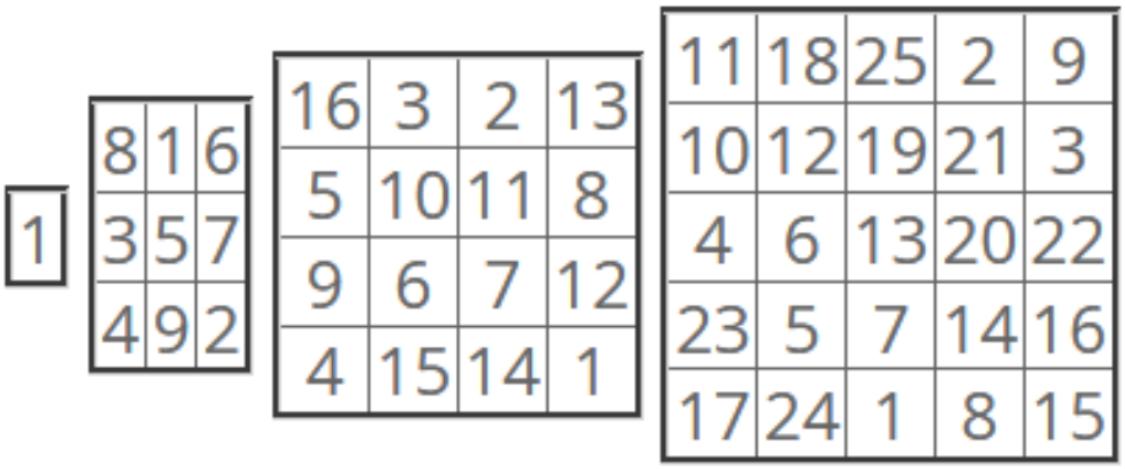

Figure 1: Esempi di quadrati magici

Esercizio 4: creare quadrati magici

Scrivere il programma `printPerfectMagicSquare.py` che generi e visualizzi un "quadrato magico perfetto" $n \times n$ con n numero DISPARI fornito dall'utente (la generazione di quadrati con n pari è molto più complessa). Realizzare l'algoritmo seguente (descritto mediante "pseudocodice"):

```
riga = n - 1
colonna = n // 2
for each k in 1, 2, ..., n*n
    scrivi k nella posizione [riga][colonna]
    incrementa riga e colonna
    se riga == n, allora riga = 0
    se colonna == n, allora colonna = 0
    se la posizione [riga][colonna] non è vuota, allora:
        ripristina riga e colonna ai loro valori precedenti
        decrementa riga
```

La richiesta iniziale del valore di n deve avvenire senza visualizzare nessun messaggio all'utente (che, quindi, deve essere un "utente informato"…) e il programma non deve visualizzare alcunché oltre alla matrice di numeri. In questo modo, i dati prodotti in uscita da questo programma possono essere forniti in ingresso al programma precedente (`isPerfectMagicSquare.py`), che può così essere utilizzato per collaudare questo usando la redirezione di output (prima) e di input (poi). Fare questi collaudi con tutti i numeri dispari fino a 19. Esempio (l'utente scrive 5):

```
C:\Users\Desktop\Python>python3 printPerfectMagicSquare.py > magicsquare.txt
5
C:\Users\Desktop\Python>python3 isPerfectMagicSquare.py < magicsquare.txt
11 18 25 2 9
10 12 19 21 3
4 6 13 20 22
23 5 7 14 16
17 24 1 8 15
OK
```

Esercizio 5: fattorizzazione in numeri primi

Scrivere il programma `factoring.py` che scomponga un numero intero maggiore di 1 nei suoi fattori primi. Il risultato della moltiplicazione di tutti (e soli) i fattori primi visualizzati deve essere uguale al numero che si doveva scomporre, quindi eventuali fattori primi ripetuti devono essere visualizzati più volte, con la molteplicità corretta (es. il numero 300 si scompone in 2, 2, 3, 5, 5)