Laboratorio 5

Elementi di Informatica e Programmazione

Lab 5 – Es 1 (1/2)

Scrivere il programma **isPerfectMagicSquare** che verifichi se un quadrato di numeri interi è un "quadrato magico".

Una disposizione bidimensionale di numeri tutti diversi avente dimensione *n* x *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). Esempi di quadrati magici perfetti:

1

8	1	6
3	5	7
4	9	2

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

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

Lab 5 – Es 1 (2/2)

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 n^2, 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 qui riportati.

Lab 5 – Es 2 (1/2)

Scrivere il programma **printPerfectMagicSquare** che generi e visualizzi un "quadrato magico perfetto" *n* x *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
```

Lab 5 – Es 2 (2/2)

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**), 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
```

Lab 5 – Es 3 (1/2)

Il *Crivello di Eratostene* è un noto algoritmo per la ricerca dei numeri primi minori di un certo valore massimo **MAX**, ed è così specificato:

- si predispone una lista di **MAX** valori ***booleani***
- ogni elemento dell'array "rappresenta" il numero intero corrispondente al proprio indice nella lista: se e solo se l'elemento è **true**, allora il numero corrispondente è stato ELIMINATO dall'insieme dei numeri primi, cioè non è un numero primo
- all'inizio si suppone che tutti i numeri siano primi; successivamente, si considera ciascun numero intero maggiore di uno, in ordine crescente, e si eliminano tutti i numeri che ne sono multipli, contrassegnandoli opportunamente nell'array (ad esempio, al primo passaggio vengono eliminati tutti i numeri pari; al secondo passaggio vengono eliminati tutti i multipli di 3, e così via
- al termine, i numeri rimasti sono tutti e soli i numeri primi cercati, non essendo multipli di alcun numero

Lab 5 – Es 3 (2/2)

Scrivere il programma **eratostene.py** che utilizzi il Crivello di Eratostene per identificare i numeri primi minori di un valore (intero positivo) **MAX** fornito dall'utente.

- **Si ricorda che 1 non è un numero primo.**
- Verificare il corretto funzionamento del programma con:
- **MAX** = 1 (non viene visualizzato nessun numero, perché non esiste nessun numero primo minore di 1)
- **MAX** = 2 (non viene visualizzato nessun numero, perché non esiste nessun numero primo minore di 2)
- **MAX** = 3 (viene visualizzato soltanto il numero 2)
- **MAX** = 4 e **MAX** = 5 (vengono visualizzati soltanto i numeri 2 e 3)
- **MAX** = 6 (vengono visualizzati soltanto i numeri 2, 3 e 5)

Lab 5 – Es 4

Scrivere il programma **longestSubstring.py** che identifichi e visualizzi la più lunga sottostringa comune a due stringhe ricevute, una dopo l'altra, come righe intere.

Esempio: la più lunga sequenza di caratteri consecutivi (cioè sottostringa) comune alle due stringhe:

```
PippoPluto2Paperino MinnieBambito2Dumbo
```

è `to2`

Verificare che il programma gestisca correttamente la situazione in cui le due stringhe non hanno alcun carattere in comune e, quindi, la più lunga sottostringa appartenente a entrambe è la stringa vuota.

Provate due strategie:

1. Il programma **non deve utilizzare** metodi che operano su stringhe, né l'operatore in.
2. Il programma **utilizza** metodi che operano su stringhe e l'operatore in.

Inutile (?) ribadire che non ha senso iniziare a scrivere codice senza aver prima individuato un algoritmo che funzioni "sulla carta"...

Attenzione: le specifiche non sono complete... Cosa deve visualizzare il programma nel caso in cui le stringhe da elaborare siano PippoPluto e PippoMinniePluto ?

Fare ipotesi ragionevoli...

Lab 5 – Es 5

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)