=== Filtro ===
Scrivere un programma in Go (il file deve chiamarsi filtro.go) che legge da standard input una stringa che contiene un solo simbolo della tastiera; se il simbolo è L, T o Z, il programma stampa la lettere disegnata con asterischi (vedi sotto), altrimenti stampa il messaggio "input non valido" (virgolette escluse).

Esempi
```
$ go run filtro.go
L
*
*
*
*
*****

$ go run filtro.go
T
*****
   *
   *
   *
   *

$ go run filtro.go
Z
*****
    *
  *
 *
*****

$ go run filtro.go
t
input non valido
```

=== Isogrammi ===
Un isogramma è una parola o una frase senza caratteri che si ripetono, tranne spazi e trattini ('-') che possono apparire più volte.

Esempi di isogrammi sono:
- velocista
- tavoli
- conquista
- break-time
- dove si cena?
- come si fa?
- isogram
- six-pack

Non sono invece isogrammi:

casa
tavolo
cruscotto
questo no
twelve-pack

Scrivere un programma in Go (il file deve chiamarsi 'isogrammi.go') dotato di:

- una funzione 'isIsogram(s string) bool' che restituisce true se la stringa s è un isogramma (ha tutti i caratteri diversi, tranne eventuali spazi e trattini ('-')

- una funzione main() che legge un numero arbitrario di righe di testo da standard input, per ciascuna riga stabilisce se la riga stessa è un isogramma o no e stampa su standard output la riga seguita da SI/NO.

N.B. per segnalare al programma la fine dell'input, dare da tastiera invio seguito da ctrl D

Esempio di esecuzione
$ go run isogrammi.go
velocista
velocista: SI
break-time
break-time: NO
six-pack
six-pack: SI
hold-me-back
hold-me-back: SI
cruscotto
cruscotto: NO
isogram
isogram: SI

Nota Bene, se lo si lancia però dandogli un input redirezionato NON si vedono gli input, ad es.:
$ ./isogrammi < uno.input
velocista: SI
tavoli: SI
supercalifragilistichespiralidoso: NO
The Quick Brown Fox Jumps Over The Lazy Dog: NO
conquista: SI
break-time: NO
dove si cena?: NO
come si fa?: SI
isogram: SI
six-pack: SI
casa: NO
tavolo: NO
cruscotto: NO
diavolo: NO

=== Luhn ===
Scrivere un programma in Go (il file deve chiamarsi 'luhn.go') che, data su linea di comando una lista di lunghezza arbitraria di numeri di carta di credito, ristampa ogni numero originale seguito da "valido" / "non valido" (virgolette escluse).

Se non sono presenti dati sulla linea di comando, il programma stampa "nessun input".

Per dati non numerici o valori con un numero di cifre diverso da 16, il programma stampa "dato non valido"

L'algoritmo di Luhn, che consente di controllare la validità di vari numeri, tra cui quelli delle carte di credito, si basa su tre passi:
- moltiplicare per due una cifra sì e una no, partendo dalla penultima cifra a destra e proseguendo verso sinistra.
- Dove la moltiplicazione ha dato un risultato a due cifre, sottrarre 9 dal prodotto
- Sommare tutte le cifre, sia quelle che si trovano in posizione pari, sia quelle che si trovano in posizione dispari
Se la somma complessiva è divisibile per 10 (la divisione non ha resto) il numero è valido.

Ad esempio, dato il numero (senza spazi)
4539 1488 0343 6467
si dovranno raddoppiare
4_3_ 1_8_ 0_4_ 6_6_

ottenendo:
8_6_ 2_7_ 0_8_ 3_3_

Sommando ora tutte le cifre
8569 2478 0383 3437

8+5+6+9+2+4+7+8+0+3+8+3+3+4+3+7 = 80

si ottiene un multiplo di 10, quindi questo numero è valido.

Esempi
$ go run luhn.go 4539148803436467 8273123273520569 4024007190270651 4929876177122565 bastiancontrario 492987
4539148803436467 valido
8273123273520569 non valido
4024007190270651 valido
4929876177122565 non valido
bastiancontrario dato non valido
492987 dato non valido

$ go run luhn.go
nessun input

Presenze
Scrivere un programma in Go (il file deve chiamarsi 'presenze.go') dotato di

- una struttura Studente (nome, cognome, matr, presenze), dove:

nome e cognome sono di tipo string
matr e presenze di tipo int

- funzione incrementaPresenze(p_studente *Studente)
che incrementa di 1 le presenze di studente

- una funzione main() che legge il nome di un file e un carattere da linea di comando

Il file conterrà una serie di righe, ciascuna con Nome, Cognome, Matr di uno studente, numero di presenze (nomi e cognomi di una parola sola e con iniziali in A-Z, a-z)

Il programma deve creare una slice di studenti con i dati del file.

Il programma deve poi incrementare le presenze degli studenti della slice
il cui nome inizia con il carattere letto (secondo parametro linea di comando)
e produrre in output la lista degli studenti con la situazione delle presenze aggiornata.

Esempio
Dato un file "uno.input" contenente:

Ida Unruh 381732 2
Carolynn Wimmer 93824923 6
Vada Furrow 28391 5
Maurine Even 28318 9
Lamar Tapp 3261725631 8
Sook Earle 763512 6
Denita Strothers 217632187 7
Carin Poteet 27651 7
Ina Blessing 29319812 5
Tish Billingsly 28738213 8
Felice Hermanson 5968598659 6
Everette Zeman 73627632 7
Agnus Birnbaum 29329832 6
Dania Crail 2329832 1
Violet Carnell 29387287 4
Marcelina Tuck 2736276 5
Freda Leonardo 2387872 7
Vina Clermont 832323 6
Gidget Phung 38723872 9
Ignacia Schaffner 23762372 4

L'esecuzione di:
$ go run presenze.go datiEsPresenze.input I

genererà:
Ida Unruh 381732 3
Carolynn Wimmer 93824923 6
Vada Furrow 28391 5
Maurine Even 28318 9
Lamar Tapp 3261725631 8
Sook Earle 763512 6
Denita Strothers 217632187 7

Carin Poteet 27651 7
Ina Blessing 29319812 6
Tish Billingsly 28738213 8
Felice Hermanson 5968598659 6
Everette Zeman 73627632 7
Agnus Birnbaum 29329832 6
Dania Crail 2329832 1
Violet Carnell 29387287 4
Marcelina Tuck 2736276 5
Freda Leonardo 2387872 7
Vina Clermont 832323 6
Gidget Phung 38723872 9
Ignacia Schaffner 23762372 5