ESERCIZIO FILTRO
Scrivere un programma che legga da riga di comando una parola contenente caratteri Unicode produca in output la stampa della parola ruotatat come nelll'esempio di esecuzione.

```
Esempio d'esecuzione:
$ go run esercizio_filtro.go filtro
filtro
iltrof
ltrofi
trofil
rofilt
ofiltr

$ go run esercizio_filtro.go maèsçòa
maèsçòa
aèsçòam
èsçòama
sçòamaè
çòamaès
òamaèsç
amaèsçò
```

ESERCIZIO 1
Scrivere un programma che legga da standard input una stringa formata da caratteri arbitrari (nel sistema di codifica Unicode) e stampi su standard output tutte le sotto stringhe della stringa data che siano costituite da soli caratteri ASCII e che abbiano la lunghezza massima tra tutte.
A tale fine definire la funzione `OnlyAscii(s string) bool` che riceve da input una stringa e restituisce `true` se questa è costituita da soli caratteri ASCII o `false` altrimenti.

```
Esempio d'esecuzione:

$ go run esercizio_1.go
Inserire input: cortaËluuuunga
luuuunga

$ go run esercizio_1.go
Inserire input: abĀdabÓm!
dab

$ go run esercizio_1.go
Inserire input: LèLettéreÀccentàteNonSònoAscií
teNonS
noAsci

$ go run esercizio_1.go
Inserire input: çÂë

$ go run esercizio_1.go
Inserire input: 61**.—;!*
61**.—;!*
```

ESERCIZIO 2
Solitamente, un numero romano presenta i simboli posizionati in ordine di valore, dal più grande al più piccolo. Il valore di un numero romano si calcola solitamente per somma del valore dei simboli che lo compongono. Per esempio, 2 equivale a II in numero romano (I+I).12 è scritto come XII, che è semplicemente X+II. In modo simile, 27 è scritto come XXVII, chi è XX + V + II. Tuttavia uno

stesso simbolo non può essere ripetuto più di 3 volte, infatti IIII non è un numero romano ben formattato. Per esprimere il valore 4, la giusta sintassi e IV pertanto, quando un simbolo con valore minore si presenta prima di un simbolo con valore maggiore, il valore più piccolo deve essere sottratto al valore più grande.
Esempi:
- IV = V - I = 5 - 1 = 4
- MCMXLIV = M + (M - C) + (L - X) + (V - X) = 1000 + (1000 - 100) + (50 - 10) +5 - 1) = 1944

Il simbolo I può comparire alla sinistra dei simboli V e X, ma una sola volta.ad esempio X equivale a 9. IX non è un numero romano ben formattato, in quanto il numero romano corrispondente a 8 e VIII. Il simbolo X può comparire alla sinistra dei simboli L e C, ma una sola volta. Ad esempio il numero 99 corrisponde a XCX: la somma di XC (90) e IX (nove). Il numero romano iC non è ben formattato infine il simbolo ci può comparire alla sinistra dei simboli D e M ma una sola volta. Ad esempio, il numero 900 corrisponde a CM. I simboli V, L e D non possono mai essere posizionati alla sinistra di simboli con valore più grande (VC ad esempio non è ben formattato in quanto il numero 95 corrisponde a XCV: XC + V).

Si scrive un programma che legga da riga di comando un numero romano (tipo string) e che stampi il suo valore in formato decimale. A tal fine si implementi la funzione `Romano2decimale (n string) int` che prende in input un numero romano ben formattato (tipo string) e restituisce il valore corrispondente in formato decimale (tipo int). Si assume che il numero romano inserito da riga di comando sia ben formattato. Si può inoltre assumere che la stringa in input sarà composta da soli caratteri ASCII.

`Esempio d'esecuzione:`

```
$ go run esercizio_2.go CXLI
141

$ go run esercizio_2.go MCMLXXXIII
1983

$ go run esercizio_2.go CCCLXXXIX
389

$ go run esercizio_2.go MMMCMXCIX
3999

$ go run esercizio_2.go MDCCLXXXIV
1784
```

ESERCIZIO 3
Scrivere un programma che:
- legge da standard input una sequenza di righe di testo;
- termina la lettura quando, premendo la combinazione di tasti Ctrl+D , viene inserito da standard input l'indicatore End-Of-File (EOF).

Ogni riga di testo è nel formato: codice;nome_prodotto;quantità
Il codice e la quantità sono formate esclusicavamente da cifre numeriche.

Definire:
- `Prodotto`: contenente i parametri nome e codice, nei quali saranno memorizzati i rispettivi dati del prodotto
- `Magazzino`: che dovrà associare ciascun prodotto inserito con la quantità presente nel magazzino

Il programma deve implementare almeno le seguenti funzioni:

- `NuovoMagazzino() Magazzino` che dovrà inizializzare e restituire una nuova variabile magazzino vuoto
- `StringProdotto(p Prodotto) string` che riceve un prodotto e restituisce una stringa formattata che lo descrive esempio:
  Codice: 4765667, Prodotto: cuscino
- `CreaProdotto(nome string, codice string) Prodotto` che dati gli estremi di un prodotto crea e restituisce una nuova variabile prodotto
- `NumeroProdotti (m Magazzino) int` che dato il magazzino restituisce il numero di prodotti in esso contenuto (0 se il magazzino è vuoto)
- `Quantità(m Magazzino, p Prodotto) int` che dati il magazzino ed un prodotto restituisce il numero di occorrenze del prodotto in magazzino
- `ModificaProdotto (m Magazzino, p Prodotto, variazione int) (magazzino, bool)` che dovrà modificare la quantità del prodotto p nel magazzino m, applicando la variazione specificata e restituire il magazzino così modificato. Inoltre dovrà essere restituito un valore booleano che specifica se l'operazione è valida nello specifico:
  - Se il prodotto non è presente in magazzino e la variazione risulta positiva, allora il prodotto viene inserito in magazzino
  - Se la quantità del prodotto dopo la variazione risultasse positiva, allora la modifica sarà applicata e l'operazione sarà valida
  - Se la quantità del prodotto dopo la variazione risultasse zero, allora il prodotto dovrà essere eliminato dal magazzino e l'operazione sarà valida
  - Se la quantità del prodotto dopo la variazione risultasse negativa, allora la modifica non dovrà essere applicata. La funzione dovrà restituire il magazzino non modificato e l'operazione non sarà valida

Il programma deve stampare su standard output:
- se tutte le operazioni di modifica sono valide, il numero di prodotti presenti in magazzino e l'elenco finale dei nomi dei prodotti con le corrispondenti quantità in magazzino ordinate alfabeticamente in modo crescente secondo il codice prodotto
- se una qualsiasi riga letta introduce un'operazione di modifica non valida il numero della riga alla quale si è verificato l'errore, il nome del prodotto la quantità del prodotto in magazzino e la variazione richiesta che ha causato l'errore.

```
Esempio d'esecuzione:
$ cat prod1.txt
47656670;cuscino;31
49328402;tv;26
60354989;tappeto;32
78549203;sedia;39
46376653;quadro;100
87660854;divano;1
47656670;cuscino;3
49328402;tv;-10
60354989;tappeto;32
78549203;sedia;-39
46376653;quadro;100

$ go run esercizio_3.go < prod1.txt
Il magazzino contiene 5 prodotti.
Codice: 46376653, Prodotto: quadro, Quantità: 200
Codice: 47656670, Prodotto: cuscino, Quantità: 34
Codice: 49328402, Prodotto: tv, Quantità: 16
Codice: 60354989, Prodotto: tappeto, Quantità: 64
Codice: 87660854, Prodotto: divano, Quantità: 1

$ cat prod2.txt
123;computer;200
```

```
500;mouse;198
2;keyboard;37
15;speaker;34
99;webcam;0
123;computer;-159
123;computer;-46
2;keyboard;-4
22;charger;12

$ go run esercizio_3.go < prod2.txt
Errore alla riga 7, Codice: 123, Prodotto: computer, Quantità: 41,
Richiesta: -46
```