Esercizio filtro – Numeri triangolari e quadrangolari

In matematica, un ***numero triangolare*** è un numero poligonale rappresentabile in forma di **triangolo**. Visivamente, preso un insieme con cardinalità pari al numero in oggetto, è possibile disporre i suoi elementi su una griglia regolare, in modo da formare un triangolo equilatero o un triangolo isoscele, come mostrato nella figura sotto per i primi 5 numeri triangolari (1, 3, 6, 10, 15).

```
1  ->   O

3  ->   O
        OO

6  ->   O
        OO
        OOO

10 ->   O
        OO
        OOO
        OOOO

15 ->   O
        OO
        OOO
        OOOO
        OOOOO
```

A pensarci bene, risulta chiaro che il calcolo del n-esimo numero triangolare, $\boldsymbol{tr}(\boldsymbol{n})$, segue la seguente regola:

$$tr(n) = \begin{cases} 0 & sse\ n = 0 \\ 1 & sse\ n = 1 \\ \sum_{i=1}^{n} i & sse\ n > 1 \end{cases}$$

Una definizione analoga, vale per i ***numeri quadrati***: un ***numero quadrato*** è un numero poligonale rappresentabile in forma di **quadrato**. Visivamente, preso un insieme con cardinalità pari al numero in oggetto, è possibile disporre i suoi elementi su una griglia regolare, in modo da formare un, come mostrato nella figura sotto per i primi 5 numeri quadrati.

```
1  ->   O

4  ->   OO
        OO

9  ->   OOO
        OOO
        OOO

16 ->   OOOO
        OOOO
        OOOO
        OOOO
```

25 ->
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO

Osservando bene, si nota che l'n-esimo numero quafrato, $\boldsymbol{qd(n)}$, è pari a:

$$\begin{cases} 0 & sse\ n = 0 \\ 1 & sse\ n = 1 \\ qd(n) = tr(n) + tr(n-1) & sse\ n > 1 \end{cases}$$

Esistono numeri triangolari che sono anche numeri quadrati.

Scrivere un programma che legga un intero ***t*** da linea di comando e quindi:

- elenchi i primi ***t*** numeri triangolari;
- elenchi i primi ***t*** numeri quadrati;
- elenchi i numeri che sono sia triangolari che quadrati.

Esempio di esecuzione

Eseguendo
```
>go run SimEsFiltro.go 11
```

si ottiene:
```
[0 1 3 6 10 15 21 28 36 45 55]
[0 1 4 9 16 25 36 49 64 81 100]
0 triangolare e quadrato
1 triangolare e quadrato
36 triangolare e quadrato
```

```
>go run SimEsFiltro.go 23
[0 1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136 153 171 190 210 231 253]
[0 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400 441 484]
0 triangolare e quadrato
1 triangolare e quadrato
36 triangolare e quadrato
```

Esercizio 1 – Inversione in stringhe

Scrivere un programma che legge da standard input due stringhe **base** e **inverti** (ciascuna stringa su un'unica riga). Entrame le stringhe sono composte da parole (una parola è una lista di caratteri delimitata da spazi).

Il programma ristampa la stringa ***base*** modificata come segue.

Il programma considera tutte le parole contenute in ***inverti*** e sostituisce ogni loro occorrenza in ***base*** con le stesse parole in cui i caratteri sono però disposti in ordine inverso (ad es.: "`abcdefg`" in ***base*** potrebbe essere sostituita con la stringa "`gfedcba`").

Data una stringa in cui sono presenti degli spazi, un possibile modo per identificare le parole presenti nella stringa è quello di utilizzare la funzione `strings.Split`.

A scopo dimostrativo si consideri il seguente esempio.

```
package main
import (
        "fmt"
        "strings"
)

func main() {
        str := "Prova di utilizzo"
        sl := strings.Split(str[:]," ") //sl è una slice di stringhe
        for _, v := range sl {
                fmt.Println(v)
        }
}
```

Output:

```
Prova
di
utilizzo
```

Esempi di esecuzione

```
>go run SimEs1.go
Inserisci stringa base:
pippo pluto minnie io lei
Inserisci stringa inverti:
io lei
Stringa risultato:
pippo pluto minnie oi iel

>go run SimEs1.go
Inserisci stringa base:
io noi voi essi
Inserisci stringa inverti:
io noi voi essi mamma
Stringa risultato:
oi ion iov isse

>go run SimEs1.go
Inserisci stringa base:
```

```
pioggia
Inserisci stringa inverti:
pio
Stringa risultato:
pioggia
```

Esercizio 2 – Triplette ordinate

Scrivere un programma che legga da standard input una sequenza di ***n*** numeri interi strettamente positivi (un numero ***x*** è strettamente positivo se è ***x>0***) separati da spazi. Il programma stampa a video i numeri letti e le terne di numeri ***[x,y,z]*** contenute nella lista tali che:

1. ***x < y < z***
2. le posizioni ***ind(x), ind(y), ind(z)*** dei numeri ***x***, ***y*** e ***z*** nella sequenza soddisfino la condizione ***ind(x) < ind(y) < ind(z).***

Per convertire una stringa che rappresenta un numero intero nel corrispondente valore intero è possibile utilizzare la funzione `strconv.Atoi`.

```
str := "9"
num, err := strconv.Atoi(str)
```

Esempi di esecuzione

```
>go run SimEs2.go
1 4 2 -3 0 0 3 5 1 0
Valori inseriti:
[1 4 2 3 5 1]
Triplette ordinate:
[1<4<5]
[1<2<3]
[1<2<5]
[1<3<5]
[2<3<5]

>go run SimEs2.go
34 1 4 7 23
Valori inseriti:
[34 1 4 7 23]
Triplette ordinate:
[1<4<7]
[1<4<23]
[1<7<23]
[4<7<23]
```

Esercizio 3 – Rombi di lettere

Scrivere un programma che legga da linea di comando un intero ***n>=0*** e quindi stampi un rombo di altezza ***2n-1*** composto da lettere maiuscole, come indicato negli esempi.

Se ***n<0*** il programma deve stampare un messaggio di errore, ovvero la stringa

`"Altezza del rombo minore di zero."`

Se ***n=0*** non viene stampato alcun carattere.

Se ***n=1***

```
A
```

Se ***n=2***

```
  A
B A B
  A
```

Se ***n=3***

```
    A
  B A B
C B A B C
  B A B
    A
```

Se ***n=4***

```
      A
    B A B
  C B A B C
D C B A B C D
  C B A B C
    B A B
      A
```

Se ***n=11***

```
                    A
                  B A B
                C B A B C
              D C B A B C D
            E D C B A B C D E
          F E D C B A B C D E F
        G F E D C B A B C D E F G
      H G F E D C B A B C D E F G H
    I H G F E D C B A B C D E F G H I
  J I H G F E D C B A B C D E F G H I J
K J I H G F E D C B A B C D E F G H I J K
  J I H G F E D C B A B C D E F G H I J
    I H G F E D C B A B C D E F G H I
      H G F E D C B A B C D E F G H
        G F E D C B A B C D E F G
          F E D C B A B C D E F
            E D C B A B C D E
              D C B A B C D
                C B A B C
                  B A B
                    A
```

Notate che:

- ogni lettera è separata dalle altre da uno spazio;
- la riga centrale, cioè quella più lunga, comincia senza alcuno spazio;
- ogni riga termina con una lettera, senza spazi aggiuntivi.

Esercizio 4 – Numero migliore

Scrivere un programma che riceve in input come argomento del main due numeri interi positivi (che chiameremo **n** e **d**); il programma deve calcolare e stampare il più piccolo numero ottenibile rimuovendo **d** cifre decimali da **n**.

Se **d** è maggiore o uguale al numero di cifre di **n**, il programma deve stampare il valore **float64 NaN** (ottenibile con una funzione del package `math`).

Per esempio, si consideri n = 4567 e d = 2. Il più piccolo numero ottenibile rimuovendo 2 cifre è 45. Altre trasformazioni possibili potrebbero essere 46, 47, 56 e 57, e 67. Notate che non è possibile considerare permutazioni delle cifre, quindi 76 non è una soluzione ammissibile.

Una possibile idea di partenza per risolvere l'esercizio è la seguente.

Si consideri un numero **n** formato da 4 cifre decimali (ad es. 4567). Si consideri una sequenza **s** di cifre binarie di lunghezza 4 (ad es. 1011). Si può paragonare la sequenza **s** ad una maschera che nasconde le cifre decimali in **n** nella stessa posizione in cui compare uno 0 in **s** (ad es. se **n** = 4567 e **s** = 1011, la cifra decimale nascosta è 5). Per ogni sequenza **s** di cifre binarie, di lunghezza pari al numero di cifre decimali che formano **n**, è possibile ricavare da **n** un nuovo numero formato dalle cifre decimali presenti in **n** e non nascoste da **s** (ad es. se **n** = 4567 e **s** = 1011, il nuovo numero che si ricava è 467).

Se **n** = 4567 e **d** = 2, le maschera **s'** = 1011 e **s''** = 0001 non portano alla corretta definizione di numeri tra cui ricercare il più piccolo numero ottenibile rimuovendo **d** cifre decimali da **n**.

Si consideri poi il seguente codice.

```
package main

import (
        "fmt"
        "strconv"
)

func main() {
        i := 5
        b_Of_i := strconv.FormatInt(int64(i),2)
        fmt.Println(b_Of_i)

        size := 8
        format := "%0" + strconv.Itoa(size) + "b"
        b_Of_i_with_LeadingZero := fmt.Sprintf(format,i)
        fmt.Println(b_Of_i_with_LeadingZero)
}
```

Output:

```
101
00000101
```

Esempi di esecuzione

```
>go run SimEs4.go 4567 2
numero migliore: 45

>go run SimEs4.go 32751960 3
numero migliore: 21960

>go run SimEs4.go 327 9
numero migliore: NaN

>go run SimEs4.go 3275 3
numero migliore: 2

>go run SimEs4.go 32175 2
numero migliore: 175
```