Esercizio 1

Scrivere un programma che legga da **standard input** una stringa di caratteri e controlli se (la stringa letta) può rappresentare una password ben definita.

Si assuma che, chiaramente, nessun carattere nella stringa può rappresentare un carattere di spaziatura, ossia un carattere il cui codice Unicode, passato come argomento alla funzione `func IsSpace(r rune) bool` del package `unicode`, fa restituire `true` alla funzione.

Una password è ben definita se, considerando la stringa di caratteri che la rappresenta, vengono soddisfatte le seguenti condizioni:

1. la stringa deve avere una lunghezza minima di 6 caratteri e una lunghezza massima di 15 caratteri;
2. almeno 2 caratteri nella stringa devono rappresentare delle lettere minuscole;
3. al massimo 3 caratteri nella stringa possono rappresentare delle lettere maiuscole;
4. almeno 2 caratteri nella stringa devono rappresentare delle cifre decimali;
5. almeno 2 caratteri nella stringa non devono rappresentare lettere o cifre decimali.

Un carattere rappresenta una lettera se il relativo codice Unicode, passato come argomento alla funzione `func IsLetter(r rune) bool` del package `unicode`, fa restituire `true` alla funzione.

Un carattere rappresenta una cifra decimale se il relativo codice Unicode, passato come argomento alla funzione `func IsDigit(r rune) bool` del package `unicode`, fa restituire `true` alla funzione.

Nel caso in cui la stringa letta rappresenti una password ben definita, il programma deve stampare:

`La pw è ben definita!`

In caso contrario, il programma deve stampare:

`La pw non è definita correttamente:`

ed uno o più dei seguenti messaggi opzionali:

`- La pw deve avere una lunghezza minima di 6 caratteri ed una lunghezza massima di 15 caratteri`

(da stampare solo se la condizione 1 non è stata soddisfatta dalla stringa letta)

`- Almeno 2 caratteri della pw devono rappresentare delle lettere minuscole`

(da stampare solo se la condizione 2 non è stata soddisfatta)

`- Al massimo 3 caratteri della pw possono rappresentare delle lettere maiuscole`

(da stampare solo se la condizione 3 non è stata soddisfatta)

`- Almeno 2 caratteri della pw devono rappresentare delle cifre decimali`

(da stampare solo se la condizione 4 non è stata soddisfatta)

`- Almeno 2 caratteri della pw non devono rappresentare lettere o cifre decimali`

(da stampare solo se la condizione 5 non è stata soddisfatta)

Esempio d'esecuzione:

```
$ go run esercizio_1.go
pa55W0rdlung4
La pw non è definita correttamente:
- Almeno 2 caratteri della pw non devono rappresentare lettere o cifre decimali

$ go run esercizio_1.go
```

```
Qu35t@_Pa55w0rD
La pw è ben definita!

$ go run esercizio_1.go
pW5
La pw non è definita correttamente:
- La pw deve avere una lunghezza minima di 6 caratteri ed una lunghezza massima di 15 car
- Almeno 2 caratteri della pw devono rappresentare delle lettere minuscole
- Almeno 2 caratteri della pw devono rappresentare delle cifre decimali
- Almeno 2 caratteri della pw non devono rappresentare lettere o cifre decimali

$ go run esercizio_1.go
PAS5WORD_LUNGA
La pw non è definita correttamente:
- Almeno 2 caratteri della pw devono rappresentare delle lettere minuscole
- Al massimo 3 caratteri della pw possono rappresentare delle lettere maiuscole
- Almeno 2 caratteri della pw devono rappresentare delle cifre decimali
- Almeno 2 caratteri della pw non devono rappresentare lettere o cifre decimali

$ go run esercizio_1.go
password
La pw non è definita correttamente:
- Almeno 2 caratteri della pw devono rappresentare delle cifre decimali
- Almeno 2 caratteri della pw non devono rappresentare lettere o cifre decimali
```

Esercizio 2

Scrivere un programma che legga da **riga di comando** due numeri naturali (interi positivi), rispettivamente `N` e `k`.

Il programma deve stampare a video tutti i numeri naturali *dispari* ottenibili rimuovendo `k` cifre decimali *consecutive* da `N`.

Oltre alla funzione `main()`, devono essere definite ed utilizzate almeno le seguenti funzioni:

- una funzione `GeneraNumeri(N, k int) []int` che riceve in input due valori `int` nei parametri `N` e `k`, e restituisce un valore `[]int` in cui sono memorizzati tutti i numeri interi positivi ottenibili rimuovendo `k` cifre decimali consecutive da `N`;
- una funzione `FiltraNumeri(sl [] int) []int` che riceve in input due valori `[]int` nel parametro `sl` e restituisce un valore `[]int` in cui sono memorizzati tutti i numeri dispari presenti in `sl`.

Si assuma che:

- i valori letti da **riga di comando** siano specificati nel formato corretto;
- il numero di cifre decimali che definiscono `N` sia maggiore di `k`.

Esempio d'esecuzione:

```
$ go run esercizio_2.go 12345 2
345
145
125
123

$ go run esercizio_2.go 1234 1
123

$ go run esercizio_2.go 12345 3
45
15

$ go run esercizio_2.go 6482 2

$ go run esercizio_2.go 9573 2
73
93
95
```

Esercizio 3

Parte 1

Come illustrato nell'immagine di seguito riportata, il piano cartesiano è diviso in quattro quadranti: I, II, III e IV quadrante.

```
                  ^
                  |
     II quadrante   |   I quadrante
             -------|------>
     III quadrante  |   IV quadrante
                  |
```

Sul piano cartesiano, ad ogni punto individuato da una coppia di numeri reali, chiamati rispettivamente ascissa e ordinata, può essere associata un'etichetta simbolica, generalmente una lettera maiuscola.

Scrivere un programma che:

- legga da **standard input** una sequenza di righe di testo;
- termini 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 una stringa che specifica l'etichetta del punto (ad es.: `A`, `B`, ...), l'ascissa e l'ordinata del punto nel seguente formato:

```
etichetta;x;y
```

> *Esempio*: Si ipotizzi che vengano inserite da standard input le seguenti di righe di testo:
>
> ```
> A;10.0;2.0
> B;11.5;3.0
> C;8.0;1.0
> D;14.0;-1.0
> ```
>
> tali righe specificano 4 punti: `A(10, 2)`, `B(11.5, 3)`, `C(8, 1)` e `D(14, -1)`.

Ogni tripla di punti letti in input definisce un triangolo che ha per vertici i punti stessi.

> *Esempio*: dati i punti dell'esempio precedente, i triangoli che possono essere definiti sono 4: `ABC`, `ABD`, `ACD` e `BCD`.

Definire:

- la struttura `Punto` per memorizzare l'`etichetta`, l'`ascissa` e l'`ordinata` di un punto sul piano cartesiano;
- la struttura `Triangolo` per memorizzare le 3 istanze di tipo `Punto` che definiscono i 3 vertici di un triangolo.

Implementare le funzioni:

- `LeggiPunti() (punti []Punto)` che:
  i. legge da **standard input** una sequenza di righe di testo nel formato `etichetta;x;y`, terminando la lettura quando viene letto l'indicatore End-Of-File (EOF);
  ii. restituisce un valore `[]Punto` nella variabile `punti` in cui è memorizzata la sequenza di istanze del tipo `Punto` inizializzate con i valori letti da **standard input**;
- `GeneraTriangoli(punti []Punto) (triangoli []Triangolo)` che riceve in input una slice di

istanze di tipo `Punto` nella variabile `punti` e restituisce una slice di istanze di tipo `Triangolo` nella variabile `triangoli` in cui sono memorizzati tutti i triangoli che possono essere generati a partire dai punti in `punti`.

Si assuma che:

- le righe di testo lette da **standard input** siano nel formato corretto;
- la tripla di valori presente in ogni riga specifichi correttamente un punto sul piano cartesiano;
- vengano lette da **standard input** almeno 3 righe di testo che specificano 3 punti distinti sul piano cartesiano.

Parte 2

La lunghezza di ciascun lato di un triangolo è pari alla distanza euclidea tra gli estremi del lato.

Per esempio, la lunghezza del lato `AB` del triangolo `ABD` è pari alla distanza euclidea tra i punti `A` e `B`: $((x_A-x_B)^2 + (y_A-y_B)^2)^{1/2}$.

Una volta terminata la fase di lettura, il programma deve stampare a video (come mostrato nell'**Esempio di esecuzione**) la descrizione del triangolo *rettangolo* definibile a partire dai punti specificati nelle righe di testo lette da **standard input** tale che:

1. uno dei due cateti del triangolo rettangolo sia parallelo all'asse delle ascisse (l'altro cateto deve essere quindi parallelo all'asse delle ordinate);
2. i vertici del triangolo rettangolo non giacciano tutti nello stesso quadrante del piano cartesiano;
3. abbia *area minore* tra tutti i triangoli rettangoli che soddisfano le condizioni 1 e 2.

> Si ricorda che è possibile calcolare l'area di un triangolo rettangolo come
> area = cateto1 *cateto2 / 2*
> *oppure con la formula di Erone*
> *area = (p* (p - l1) *(p - l2)* (p - l3))$^{1/2}$

Se non esistono triangoli rettangoli che soddisfano le condizioni 1 e 2, il programma non deve stampare nulla.

Oltre alla funzione `main()`, devono essere definite ed utilizzate almeno le seguenti funzioni:

- `Distanza(p1, p2 Punto) float64` che riceve in input due instanze del tipo `Punto` nei parametri `p1` e `p2` e restituisce un valore `float64` pari alla distanza euclidea tra i punti rappresentati da `p1` e `p2`;
- `StringPunto(p Punto) string` che riceve in input un'instanza del tipo `Punto` nel parametro `p` e restituisce un valore `string` che corrisponde alla rappresentazione `string` di `p` nel formato `etichetta = (x, y)`;
- `StringTriangolo(t Triangolo) string` che riceve in input un'instanza del tipo `Triangolo` nel parametro `t` e restituisce un valore `string` che corrisponde alla rappresentazione `string` di `t` nel formato `Triangolo rettangolo con vertici VERTICE_1, VERTICE_2 e VERTICE_3, ed area AREA.`, dove `VERTICE_1`, `VERTICE_2` e `VERTICE_3` sono le rappresentazioni `string` delle istanze del tipo `Punto` che rappresentano i vertici di `t`, mentre `AREA` è il valore `float64` che specifica l'area di `t`.

Esempio d'esecuzione:

```
$ cat punti1.txt
A;4;8
B;4;2
C;7;2
D;5;2
E;4;9
F;5;9
G;-5;2
H;-4;9
I;4;-3
L;-5;-3
M;-5;3
```

```
$ go run esercizio_3.go < punti1.txt
Triangolo rettangolo con vertici B = (4.0, 2.0), D = (5.0, 2.0) e I = (4.0, -3.0),
ed area 2.5.

$ cat punti2.txt
A;4;8
B;4;3
C;7;2
D;5;-2
E;7;7
F;-5;9
G;-5;2
H;-4;9
I;4;-3
L;5;-3
M;-5;3

$ go run esercizio_3.go < punti2.txt
Triangolo rettangolo con vertici B = (4.0, 3.0), I = (4.0, -3.0) e L = (5.0, -3.0),
ed area 3.0.

$ cat punti3.txt
A;4;8
B;5;3
C;-7;2

$ go run esercizio_3.go < punti3.txt
```