Esercizio 3

Una posizione geografica è definita dalle sue coordinate GPS. Una sequenza di posizioni definisce un percorso, che può essere usato in un sistema di navigazione, come rotta aerea, etc..

Scrivere un programma che:

- legga da **riga di comando** due valori reali: `soglia` e `distanza`;
- legga da **standard input** una sequenza di righe di testo;
- termini la lettura premendo la combinazione di tasti `Ctrl+D` (Indicatore End-Of-File).

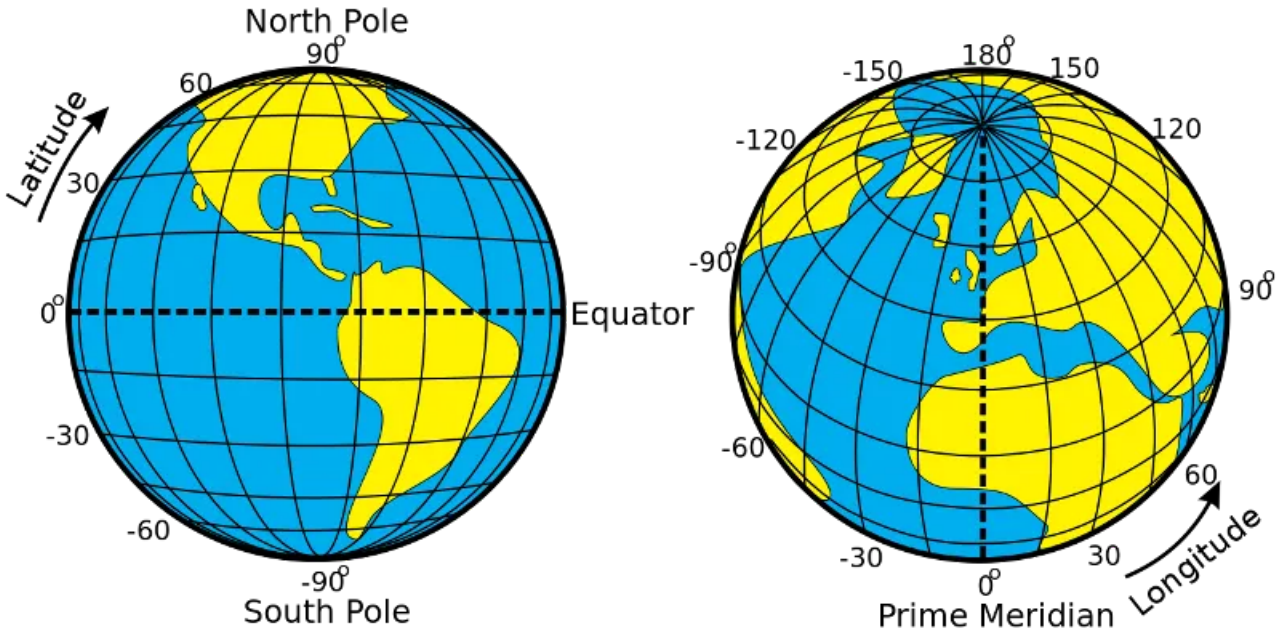
Ogni riga del testo è una stringa nel formato:

```
etichetta;latitudine;longitudine
```

La tripla di valori separati dal carattere `;` specifica una posizione geografica e si compone di:

1. *etichetta*: una stringa che specifica l'etichetta associata alla posizione (es: nome del luogo);
2. *latitudine*: un valore reale tra 90 e -90 (inclusi) che specifica la latitudine della posizione e assume il valore 0° all'equatore, +90° sul polo nord e -90° sul polo sud;
3. *longitudine*: un valore reale tra 180 (incluso) e -180 (escluso) che specifica la longitudine della posizione e assume il valore 0° al meridiano di Greenwich, con valori positivi verso est e negativi verso ovest, fino a 180° agli antipodi di Greenwich.

La seguente grafica descrive il range dei valori *latitudine* e *longitudine*:



Ad esempio, si ipotizzi che vengano inserite da **standard input** le seguenti di righe di testo:

```
Milano;45.4642;9.19
Roma;41.9028;12.4964
Napoli;40.8518;14.2681
```

Ciascuna riga specifica una posizione geografica, mentre la sequenza di righe specifica il percorso da Milano a Napoli passando per Roma.

Parte 1

Implementare:

- Il tipo `Posizione` per memorizzare l' `etichetta` , la `latitudine` e la `longitudine` di una posizione geografica.
- La funzione `NuovaPosizione(etichetta string, latitudine float64, longitudine float64) (Posizione, bool)` che riceve in input una tripla di valori relativi ad una posizione geografica e restituisce due valori: 1) l'istanza del tipo `Posizione` corrispondente; 2) il valore booleano `true` se i valori specificati nei parametri `latitudine` e `longitudine` sono validi, il valore booleano `false` altrimenti.
- La funzione `StringPosizione(posizione Posizione) string` che riceve in input un'instanza di tipo `Posizione` nel parametro `posizione` e restituisce un valore `string` nel formato `etichetta (latitudine, longitudine)` , dove `etichetta` è il valore `string` che specifica l'etichetta di `posizione` , mentre `latitudine` e `longitudine` sono i valori `float64` che specificano rispettivamente la latitudine e la longitudine di `posizione` .

Ad esempio, ricevendo in input l'istanza di `Posizione` relativa alla posizione geografica di Milano, `StringPosizione` restituisce:

```
Milano (45.4642, 9.19)
```

Parte 2

Implementare:

- La funzione `StringPercorso(percorso []Posizione) string` che riceve in input un'instanza di tipo `[]Posizione` nel parametro `percorso` e restituisce un valore `string` nel formato `Percorso da POSIZIONE_1 a POSIZIONE_2, cambi: N` , dove `POSIZIONE_1` e `POSIZIONE_2` sono le rappresentazioni `string` delle istanze di tipo `Posizione` che rappresentano gli estremi del `percorso` , mentre `N` è il numero delle posizioni intermedie (quindi esclusi gli estremi) del `percorso` .

Ad esempio, per il percorso da Milano a Napoli, `StringPercorso` restituisce:

```
Percorso da Milano (45.4642, 9.19) a Napoli (40.8518, 14.2681), cambi: 1
```

Implementare il codice che stampi tutte le sottosequenze del percorso con le seguenti caratteristiche:

- ciascun spostamento tra le posizioni della sottosequenza sia verso est, e che la differenza di longitudine sia al più di valore `distanza` .
- ciascuna posizione della sottosequenza deve avere la stessa latitudine di quella precedente, a meno del valore `soglia`

Nota: sono validi anche i percorsi che superano il meridiano con longitudine 180°

Esempio d'esecuzione:

```
$ go run esercizio_3.go 1 106 < punti1.txt
Percorso da Casablanca (33.600000, -7.616400) a Baghdad (33.338600, 44.393900), cambi: 6
Percorso da Xianyang (34.345600, 108.714700) a Ōsaka (34.750000, 135.460100), cambi: 10
Percorso da Los Angeles (34.113900, -118.406800) a Atlanta (33.762600, -84.422800), cambi: 4

$ go run esercizio_3.go 0.65 107 < punti1.txt
Percorso da Casablanca (33.600000, -7.616400) a Baghdad (33.338600, 44.393900), cambi: 6
Percorso da Xianyang (34.345600, 108.714700) a Shangqiu (34.450400, 115.650000), cambi: 2
Percorso da Suzhou (33.636100, 116.978900) a Fukuoka (33.595000, 130.410000), cambi: 5
Percorso da Hiroshima (34.387800, 132.442900) a Atlanta (33.762600, -84.422800), cambi: 7
```

```
$ go run esercizio_3.go 0.5 68 < punti2.txt
Percorso da Kansas City (39.123900, -94.554100) a Louisville (38.166200, -85.648800), cambi: 3
Percorso da Washington (38.904700, -77.016300) a Lisbon (38.722700, -9.144900), cambi: 2
Percorso da Athens (37.983300, 23.733300) a Dushanbe (38.560000, 68.773900), cambi: 4
Percorso da Baoding (38.870400, 115.480000) a Pyongyang (39.019400, 125.754700), cambi: 5
Percorso da Incheon (37.476100, 126.642200) a Seoul (37.566300, 126.999700), cambi: 2

$ go run esercizio_3.go 1 67 < punti2.txt
Percorso da San Francisco (37.756200, -122.443000) a Washington (38.904700, -77.016300), cambi: 6
Percorso da Lisbon (38.722700, -9.144900) a Pyongyang (39.019400, 125.754700), cambi: 11
Percorso da Incheon (37.476100, 126.642200) a Sendai (38.287100, 141.021700), cambi: 3
```