Esercizio 1

Scrivere un programma che legga da **riga di comando** una sequenza di valori.

Il primo valore che definisce la sequenza (da sinistra verso destra) è in posizione 0, il secondo in posizione 1, etc.

La sequenza è valida se ciascun numero intero che appare in una posizione pari all'interno della sequenza è maggiore della somma dei numeri interi che lo precedono.

Si assuma che:

- la sequenza di valori letta da **riga di comando** sia definita da almeno un valore;
- la somma dei numeri interi che precedono il primo numero intero che compare all'interno della sequenza sia uguale a `0`.

Nel caso in cui la sequenza letta sia valida, il programma deve stampare:

`Sequenza valida.`

In caso contrario, il programma deve stampare:

`Valore in posizione POSIZIONE non valido.`

dove `POSIZIONE` è la posizione del primo valore che invalida la sequenza.

Ad esempio, se la sequenza di valori letta da **riga di comando** fosse:

`2 5 saltaControllo abc 6`

il programma deve stampare:

`Valore in posizione 4 non valido.`

Esempio d'esecuzione:

```
$ go run esercizio_1.go 1 sequenza 2 valida 4 test 8
Sequenza valida.

$ go run esercizio_1.go 1 sequenza 2 non 3 valida 8
Valore in posizione 4 non valido.

$ go run esercizio_1.go 1 sequenza 2 3 valida 8
Sequenza valida.

$ go run esercizio_1.go sequenza -1 valida 4 4
Sequenza valida.

$ go run esercizio_1.go -1 sequenza non valida 4
Valore in posizione 0 non valido.
```

Esercizio 2

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

Il programma deve stampare a video tutti i numeri naturali *pari* ottenibili mantenendo `k` cifre decimali consecutive presenti in `N` ed eliminando le altre.

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 mantenendo `k` cifre decimali consecutive presenti in `N` ed eliminando le altre;
- 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 pari 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
12
34

$ go run esercizio_2.go 1234 1
2
4

$ go run esercizio_2.go 12345 3
234

$ go run esercizio_2.go 6482 2
64
48
82

$ go run esercizio_2.go 9573 2
```

Esercizio 3

Scrivere un programma che legga da **riga di comando** due stringhe di caratteri `Persone.txt` e `Legami.txt`. Ciascuna delle stringhe è il nome di un file di testo memorizzato nella stessa directory in cui è memorizzato il programma.

a) Ogni riga contenuta nel file `Persone.txt` è una stringa nel seguente formato:

id_persona;*nome*;*cognome*;*età*

I valori separati dal carattere `;` specificano una persona:

1. *id_persona*: un numero intero che identifica univocamente la persona;
2. *nome*: una stringa che specifica il nome della persona;
3. *cognome*: una stringa che specifica il cognome della persona;
4. *età*: un numero intero che specifica l'età della persona.

b) Ogni riga contenuta nel file `Legami.txt` è una stringa nel seguente formato:

id_persona_1;*id_persona_2*

La coppia di valori separati dal carattere `;` specifica un legame diretto tra la persona identificata da `id_persona_1` e la persona identificata da `id_persona_2`, e viceversa.

Per ciascuna coppia, i valori `id_persona_1` e `id_persona_2` sono diversi e, in particolare, ciascun di essi è associato ad un insieme di valori che specificano una persona nel file `Persone.txt`.

Il programma deve stampare a video la descrizione dell'insieme di 3 persone tale che:

- almeno una persona dell'insieme abbia un legame diretto con ciascuna delle altre due persone appartenenti all'insieme;
- l'età media delle persone dell'insieme sia massima.

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

- una funzione `StringPersona(p Persona) string` che riceve in input un'instanza del tipo `Persona` nel parametro `p` e restituisce un valore `string` che corrisponde alla rappresentazione `string` di `p` nel formato `IDENFICATIVO - COGNOME NOME: ETÀ`, dove `IDENTICATIVO` ed `ETÀ` sono i valori string corrispondenti ai valori interi dei campi `id` ed `età` di `p`, mentre `COGNOME` e `NOME` sono i valori `string` dei campi `cognome` e `nome` di `p`;
- una funzione `StringInsiemeTre(i InsiemeTre) string` che riceve in input un'instanza del tipo `InsiemeTre` nel parametro `i` e restituisce un valore `string` che corrisponde alla rappresentazione `string` di `i` nel formato `Insieme formato da: \n PERSONA_1;\n PERSONA_2;\n PERSONA_3.\n Età media: ETÀ_MEDIA`, dove `PERSONA_1`, `PERSONA_2` e `PERSONA_3` sono le rappresentazioni delle tre istanze del tipo `Persona` che definiscono `i`, mentre `ETÀ_MEDIA` è il valore reale del campo `etàMedia` di `i`.

Si assuma che:

- i valori letti da **riga di comando** siano specificati nel formato corretto;
- le righe di testo contenute nei file `Persone.txt` e `Legami.txt` siano nel formato corretto;
- la coppia di valori presente in ogni riga del file `Legami.txt` specifichi correttamente, ed in modo univoco, un legame tra due persone specificate nel file `Persone.txt`;
- i legami specificati nel file `Legami.txt` permettano di definire almeno un insieme di 3 persone (specificate nel file `Persone.txt`) tale che almeno una persona dell'insieme abbia un legame diretto con ciascuna delle altre due persone appartenenti all'insieme.

Esempio d'esecuzione:

```
$ go run esercizio_3.go Persone.txt Legami.txt
Insieme formato da:
 1 - Mario Rossi: 49;
 7 - Maria Rossi: 47;
 18 - Carla Bianchi: 45.
 Età media: 47.000000
```