4. il programma contiene le funzioni specificate, ma con segnatura diversa.

3 Il problema

Obiettivo del progetto è gestire un dizionario di *parole* e *schemi*.

Parole e catene di parole

Una *parola* è una sequenza finita di caratteri appartenenti all'alfabeto inglese minuscolo {a, b, c, . . . , y, z}.

Definiamo *operazioni elementari di editing* su una parola x le seguenti:

- Inserzione di un carattere in qualsiasi posizione in x. Ad esempio, "pippo" diventa "pioppo" tramite inserzione di "o".
- Cancellazione di un carattere in qualsiasi posizione in x. Ad esempio, "capra" diventa "capa" tramite cancellazione di "r".
- Sostituzione di un carattere con un altro in qualsiasi posizione in x. Ad esempio, "cane" diventa "rane" tramite la sostituzione di "c" con "r".
- Scambio della posizione di due caratteri adiacenti qualsiasi in x. Ad esempio "trota" diventa "torta" scambiando di posizione "r" con "o".

Date due parole x e y la loro *distanza di editing* è data dal minor numero di operazioni elementari di editing necessarie per passare da x a y. Ad esempio, "cavolo" e "cavallo" hanno distanza 2, in quanto si passa dalla prima alla seconda sostituendo la prima "o" con una "a" e inserendo una "l"; la distanza fra "capra" e "arpa" è 2, in quanto occorre cancellare la "c" e scambiare la posizione di "p" e "r"; la distanza fra "pesce" e "sedia" è 4 (si noti che ci sono più modi per passare da "pesce" a "sedia" con 4 operazioni elementari).

Due parole che hanno distanza di editing pari a 1 sono dette *simili*.

Una *catena* tra due parole x e y è una sequenza di parole che inizia con x, finisce con y e tale che la distanza di editing sia 1 per ogni coppia di parole consecutive nella sequenza.

Un *gruppo* è un insieme massimale di parole del dizionario che possono essere trasformate l'una nell'altra con una catena di parole tutte interne al gruppo.

Esempio Consideriamo il dizionario formato dalle parole {"bba", "aa", "aba", "aaa", "cca"}. La sequenza "aa", "aba", "bba" costituisce una catena tra "aa" e "bba". Il gruppo che contiene la parola "aa" è formato dalle parole { "aa", "aaa", "aba", "bba"}. La parola "cca" non fa parte del gruppo, infatti tra "cca" e ciascun'altra parola del dizionario non esistono catene formate solo da parole del dizionario.

Schemi

Uno *schema* è una sequenza finita di caratteri appartenenti all'alfabeto inglese {a, b, c, . . . , y, z } ∪ {A, B, C, . . . , Y, Z}, che contiene almeno una lettera maiuscola in {A, B, C, . . . , Y,Z}.

Un'*assegnazione* è una funzione σ da {A, B, C, . . . , Z} a {a, b, c, . . . , z}. In altri termini, a ogni lettera maiuscola l'assegnazione associa una lettera minuscola.

Dato uno schema $S = \alpha_1 \dots \alpha_n$ e un'assegnazione σ, denotiamo con $\sigma(S)$ la parola $\beta_1 \dots \beta_n$ tale che, per ogni $1 \leq i \leq n$,: se α_i è una lettera maiuscola allora $\beta_i = \sigma(\alpha_i)$; se α_i è una lettera minuscola allora $\beta_i = \alpha_i$.
Una parola x è *compatibile* con uno schema S se esiste un'assegnazione σ tale che $x = \sigma(S)$.

Esempio La parola "acca" è compatibile con lo schema "aBBa", con lo schema "aBCa" e con lo schema "CDcC". Ogni parola di 4 lettere è compatibile con lo schema "ABCD", mentre lo schema "ABBA" è compatibile con tutte e sole le parole palindrome di 4 lettere.

Due schemi si dicono *compatibili* se esiste almeno una parola che è compatibile con entrambi.
Due schemi S e T si dicono *legati* se esiste una sequenza di schemi, ciascuno compatibile con il successivo, in cui il primo è S e l'ultimo è T. Una *famiglia* di schemi è un insieme massimale di schemi legati tra loro.

4 Specifiche di progettazione

Si richiede di modellare la situazione con strutture di dati opportune e di progettare algoritmi che permettano di eseguire efficientemente le operazioni elencate sotto.

Le operazioni indicate con `[LUGLIO]` o `[SETTEMBRE]` devono essere considerate solo per gli appelli di luglio o settembre, rispettivamente. Le altre operazioni devono essere considerate per ogni appello.

Le scelte di modellazione e di progettazione fatte devono essere discusse nella relazione, includendo l'analisi dei costi risultanti per le diverse operazioni.

- **crea ()**

  Crea un nuovo dizionario vuoto (eliminando l'eventuale dizionario già esistente).

- **carica** ($file$)

  Inserisce nel dizionario le parole e/o gli schemi contenuti nel file di nome `file`; `file` è di tipo testo e le parole / gli schemi sono separati da uno o più caratteri di spaziatura (compresi tabulatori e newline). Se `file` non esiste non viene eseguita alcuna operazione.

- **stampa_parole ()**

  Stampa tutte le parole del dizionario.

- **stampa_schemi ()**

  Stampa tutti gli schemi del dizionario.

- **inserisci** (w)

  Inserisce nel dizionario la parola / lo schema w; se w è già presente non viene eseguita alcuna operazione.

- **elimina** (w)

  Elimina dal dizionario la parola / lo schema w; se w non è nel dizionario non viene eseguita alcuna operazione.

- **ricerca** (S)

  Stampa lo schema S e poi l'insieme di tutte le parole nel dizionario che sono compatibili con lo schema S.

- **distanza** (x, y)

  Stampa la distanza di editing fra le due parole x e y.

- **catena** (x, y)

  Stampa una catena di lunghezza minima tra x e y di parole nel dizionario. Se tale catena non esiste o se x o y non sono nel dizionario, stampa "`non esiste`".

- `[LUGLIO]` **gruppo** (x)

  Stampa il gruppo delle parole che contiene la parola x. Se x non è nel dizionario, stampa "`non esiste`".

- `[SETTEMBRE]` **famiglia** (S)

  Stampa la famiglia di schemi che contiene S. Se S non è nel dizionario, stampa "`non esiste`".

5 Specifiche di implementazione

1. Il programma deve leggere dallo standard input (`stdin`) una sequenza di linee (separate da a-capo), ciascuna delle quali corrisponde a una linea della prima colonna della Tabella 1, w, x, y e S sono stringhe sull'alfabeto inglese con S contenente almeno una maiuscola, x e y formate solo da minuscole.

   I vari elementi sulla linea sono separati da uno o più spazi. Quando una linea è letta, viene eseguita l'operazione associata; le stampe sono effettuate sullo standard output (`stdout`), e ogni operazione di stampa deve iniziare su una nuova linea.

2. Il programma deve contenere:

   - la definizione di un tipo `dizionario` che rappresenta l'intero dizionario;
   - una funzione con segnatura:
     `newDizionario () dizionario`
     che implementa l'operazione **crea ()**, ovvero che crea un nuovo dizionario, lo inizializza e lo restituisce.
   - una funzione con segnatura:
     `esegui (d dizionario, s string)`
     che applica al dizionario rappresentato da `d` tutte le altre operazioni definite dalla stringa s, secondo quanto specificato nella Tabella 1.

Nota. Non vi sono vincoli sulla lunghezza delle parole e degli schemi.

Formato di output

1. Nel caso di insiemi di parole o di schemi – ad esempio negli output di **stampa_parole ()**, **stampa_schemi ()**, **ricerca ()**, **gruppo ()**, **famiglia ()** – le parole / gli schemi devono essere visualizzati uno per riga, racchiusi da parentesi quadre. Ad esempio, l'insieme delle parole "cane", "gatto", "casa' deve essere visualizzato come segue:

	Linea di input	Operazione
	`c`	**crea** ()
	`t`	Termina l'esecuzione del programma
	`c` $file$	**carica** ($file$)
	`p`	**stampa_parole** ()
	`s`	**stampa_schemi** ()
	`i` w	**inserisci** (w)
	`e` w	**elimina** (w)
	`r` S	**ricerca** (S)
	`d` x y	**distanza** (x, y)
	`c` x y	**catena** (x, y)
`[LUGLIO]`	`g` x	**gruppo** (x)
`[SETTEMBRE]`	`f` S	**famiglia** (S)

Tabella 1: Specifiche del programma

```
[
cane
gatto
casa
]
```

L'insieme degli schemi "aBCa", "CDcC", "ABCD", "ABBA' deve essere visualizzato come segue:

```
[
aBCa
CDcC
ABCD
ABBA
]
```

L'ordine in cui appaiono parole e schemi non è rilevante.

2. L'operazione **ricerca** (S) stampa lo schema S seguito da ":" (due-punti) e dall'insieme delle parole compatibili con S racchiuse tra quadre, come nell'esempio seguente:

```
aC:[
aa
```

```
ab
]
```

Anche in questo caso l'ordine in cui appaiono le parole non è rilevante.

3. L'operazione **catena ()** stampa le parole della catena racchiuse tra parentesi tonde. Ad esempio, la catena formate dalla sequenza di parole "aa", "aba", "bba" deve essere visualizzata come segue:

```
(
aa
aba
bba
)
```

In quest'ultimo caso, l'ordine delle parole è importante!

6 Esempi di esecuzione

Inserimento di parole / schemi

Supponiamo che le linee di input siano:

```
c
i a
i b
i Aa
i aB
i a
i Aa
p
s
t
```

L'output prodotto dal programma deve essere:

```
[
a
b
]
[
aB
Aa
]
```

Elimininazione di parole/schemi

Supponiamo che le linee di input siano:

```
c
i a
i b
i Aa
i aB
e a
e Aa
p
s
t
```

L'output prodotto dal programma deve essere:

```
[
b
]
[
aB
]
```

Ricerca di schema

Supponiamo che le linee di input siano:

```
c
i aa
i ab
r aC
t
```

L'output prodotto dal programma deve essere:

```
aC:[
aa
ab
]
```

Distanza tra parole

Supponiamo che le linee di input siano:

```
c
d aa aba
d aa aa
t
```

L'output prodotto dal programma deve essere:

```
1
0
```

Catena

Supponiamo che le linee di input siano:

```
c
i aa
i aaa
i aba
i bba
c aa bba
c aa bb
c aa aa
t
```

L'output prodotto dal programma deve essere:

```
(
aa
aba
bba
)
non esiste
(
aa
)
```

Gruppo

Supponiamo che le linee di input siano:

```
c
i aa
i aba
i aaa
i cca
i bba
g aa
t
```

L'output prodotto dal programma deve essere:

```
[
aa
aaa
aba
bba
]
```

Famiglia

Supponiamo che le linee di input siano:

```
c
i ab
i bb
i aC
i Ab
i AA
i Ba
f Ab
t
```

L'output prodotto dal programma deve essere:

```
[
aC
Ab
AA
]
```

Esecuzione di test automatici

Il file allegato "`test x formato.zip`" contiene dei file per eseguire test automatici Go: si invita all'utilizzo di tali test per verificare la correttezza del formato dell'output prodotto dal programma.

Per eseguire i test Go procedere come segue:

- aprire un terminale
- spostarsi della directory del programma
- salvare il programma con nome `solution.go`
- lanciare il comando `go mod init solution`
- creare l'eseguibile con il comando `go build solution.go`
- copiare nella directory i file contenuti nell'archivio con i test
- lanciare il comando `go test -v` (e osservare l'output)