Data una frase, è interessante contare il numero di occorrenze di ciascuna singola parola che la compone. Il tipo di dato astratto `Occorrenze` deve essere in grando di memorizzare in ordine alfabetico tutte le parole distinte presenti in una frase e il numero di ripetizioni (*occorrenze*, in gergo tecnico) di ciascuna parola. Non si deve fare alcuna ipotesi sulla lunghezza massima della frase. Invece, la lunghezza massima di ciascuna parola si supponga essere 30. Per semplicità si assuma che le **parole siano sempre MAIUSCOLE**.

Implementare la classe `Occorrenze`, dotandola delle seguenti funzionalità:

--- Metodi invocati nella PRIMA PARTE di main.cpp: ---

- ✓ **`Occorrenze o(frase);`**

Costruttore, che crea un nuovo oggetto `Occorrenze` utilizzando le parole presenti nell'argomento `frase`. Si supponga che `frase` contenga solo parole maiuscole e che le parole siano separate da uno o più spazi bianchi. Si supponga inoltre che **non siano presenti spazi bianchi all'inizio della `frase`** e che la lunghezza di ciascuna parola nella `frase` **sia minore o uguale a 30**.

- ✓ **`cout << o;`**

Operatore di uscita per oggetti di tipo `Occorrenze`. Un oggetto `Occorrenze` viene stampato secondo il seguente formato:

```
CHE:2
CIEL:1
DIMMI:1
FAI:2
IN:1
LUNA:2
SILENZIOSA:1
TU:1
```

Nell'esempio di sopra le parole presenti sono 8, ripetute, rispettivamente, 2, 1, 1, 2, 1, 2, 1, ed 1 volte. Notare che le parole vanno elencate *in ordine alfabetico*. Nel caso non vi siano parole, non deve stampare nulla.

- ✓ **`o%val;`**

Operatore di modulo, che restituisce il numero di parole che compaiono almeno `val` volte all'interno di `o`.

--- Metodi invocati nella SECONDA PARTE di main.cpp: ---

- ✓ **`o+=parola;`**

Operatore di somma ed assegnamento, tra un oggetto `Occorrenze` e la C-stringa `parola` (si supponga che tale C-stringa non contenga spazi bianchi, e che quindi sia un'unica parola). L'operatore modifica l'oggetto `o` nel seguente modo: se la parola era già presente, ne incrementa le occorrenze di uno, altrimenti aggiunge tale parola, con occorrenze pari ad 1. Si assuma anche in questo caso che la lunghezza di `parola` sia minore o uguale a 30.

- ✓ **`o[parola];`**

Operatore parentesi quadra, che accetta come argomento una singola `parola` e restituisce il numero di occorrenze di quella parola all'interno dell'oggetto `o`. Qualora la parola non sia presente, deve restituire zero. Si supponga che `parola` non contenga spazi bianchi.

✓ **`o-=C;`**

Operatore di sottrazione e assegnamento, tra un oggetto `Occorrenze` ed il singolo carattere `C`. L'operatore modifica l'oggetto `o` togliendo tutte le occorrenze delle parole che iniziano per `C`. Per semplicità si può assumere che `C` sia una lettera maiuscola.

✓ **`~Occorrenze();`**

Distruttore per oggetti di tipo `Occorrenze`.

Mediante il linguaggio C++, realizzare il tipo di dato astratto **`Occorrenze`**, definito dalle precedenti specifiche. Non è permesso utilizzare funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc.

```
// file main.cpp

#include "compito.h"

int main() {
    {
        cout <<"--- PRIMA PARTE ---" << endl;

        cout << "Test del costruttore e dell'operatore di uscita" << endl;
        Occorrenze o("CHE FAI TU LUNA IN CIEL DIMMI CHE FAI SILENZIOSA LUNA");
        cout << o << endl;

        cout << "Test dell'operatore di modulo: (deve stampare 3)" << endl;
        cout << o % 2 << endl << endl;


        cout <<"--- SECONDA PARTE ---" << endl;

        cout << "Test dell'operatore di somma ed assegnamento: (LUNA:3, DAMMI:1)" << endl;
        o += "LUNA";
        o += "DAMMI";
        cout << o << endl;

        cout << "Test dell'operatore parentesi quadra:" << endl;
        cout << "-> numero di occorrenze della parola LUNA (deve stampare 3): " << o["LUNA"] << endl;
        cout << "-> numero di occorrenze della parola CHE  (deve stampare 2): " << o["CHE"] << endl <<endl;

        cout << "Test dell'operatore di sottrazione e assegnamento (rimuove DAMMI e DIMMI):" << endl;
        o -= 'D';
        cout << o << endl;

        cout << "Test del distruttore ('o' sta per essere distrutto)" << endl;
    }

}
```

USCITA ATTESA

```
--- PRIMA PARTE ---
Test del costruttore e dell'operatore di uscita
CHE:2
CIEL:1
DIMMI:1
FAI:2
IN:1
LUNA:2
SILENZIOSA:1
TU:1

Test dell'operatore di modulo: (deve stampare 3)
3

--- SECONDA PARTE ---
Test dell'operatore di somma ed assegnamento: (LUNA:3, DAMMI:1)
CHE:2
CIEL:1
DAMMI:1
DIMMI:1
FAI:2
IN:1
LUNA:3
SILENZIOSA:1
TU:1

Test dell'operatore parentesi quadra:
-> numero di occorrenze della parola LUNA (deve stampare 3): 3
-> numero di occorrenze della parola CHE  (deve stampare 2): 2

Test dell'operatore di sottrazione e assegnamento (rimuove DAMMI e DIMMI):
CHE:2
CIEL:1
FAI:2
IN:1
LUNA:3
SILENZIOSA:1
TU:1

Test del distruttore ('o' sta per essere distrutto)
```