Un `OggettiViaggio` rappresenta una lista di oggetti da ricordarsi di portare in viaggio. Ogni oggetto è caratterizzato da una descrizione, che è una stringa di al più 40 caratteri, e dall'informazione se tale oggetto è stato preso (cioè, messo in valigia) oppure no. **Non ci possono essere due oggetti con la stessa descrizione nella stessa lista**.

Implementare le seguenti operazioni che possono essere effettuate su un `OggettiViaggio`:

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

✓ **`OggettiViaggio ov;`**

Costruttore di default che inizializza un `OggettiViaggio`, inizialmente vuoto.

✓ **`cout << ov;`**

Operatore di uscita per il tipo `OggettiViaggio`, che stampa a schermo la lista di oggetti nel seguente formato:

```
- Dentifricio
- Spazzolino
X Documenti
- Caricatore
```

Gli oggetti con la "`X`" indicano quelli presi, mentre gli oggetti con il "`-`" indicano quelli ancora non presi. Gli oggetti devono essere stampati in ordine di inserimento. Per esempio, nella `OggettiViaggio` di cui sopra, prima è stato inserito "Dentifricio", poi "Spazzolino", ecc.

✓ **`ov.aggiungi(descr);`**

Metodo che aggiunge all'`OggettiViaggio ov` un nuovo oggetto con descrizione `descr`. L'oggetto è inizialmente da prendere. Se `descr` non sta in 40 caratteri, l'oggetto non viene aggiunto e l'`OggettiViaggio` rimane invariato. Idem nel caso in cui un oggetto con la stessa descrizione fosse già presente.

✓ **`ov.prendi(descr);`**

Metodo che prende (cioè, mette in valigia) l'oggetto con descrizione `descr` dell'`OggettiViaggio ov`, se non era già preso. Se un oggetto con tale descrizione non esiste, l'`OggettiViaggio` rimane invariato.

✓ **`ov.viaggia();`**

Metodo che compie il viaggio, e di conseguenza fa tornare tutti gli oggetti dell'`OggettiViaggio ov` nello stato di non presi, in attesa di organizzare il prossimo viaggio.

✓ **`OggettiViaggio ov2 = ov;`**

Costruttore di copia che inizializza un `OggettiViaggio` uguale ad `ov`.

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

- ✓ **`~OggettiViaggio();`**

Qualora sia necessario, implementare il distruttore.

- ✓ **`ov.rimuovi(descr);`**

Metodo che rimuove l'oggetto descritto da `descr` da `ov`. L'oggetto viene rimosso sia che sia stato preso sia che no. Se `descr` non è presente in `ov`, l'`OggettiViaggio` rimane inalterato.

- ✓ **`ov += ov2;`**

Operatore di somma e assegnamento tra due `OggettiViaggio`, che aggiunge a quello a sinistra tutti gli oggetti di quello a destra. Tali oggetti conservano le descrizioni e lo stato preso/non preso che avevano nell'`OggettiViaggio` a destra.

- ✓ **`!ov;`**

Operatore di negazione logica per `OggettiViaggio`, che restituisce un `OggettiViaggio` che contiene solo gli oggetti non presi in `ov`. Tali oggetti sono nello stato di non preso nell'`OggettiViaggio` risultante.

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

Uscita che deve produrre il programma

```
--- PRIMA PARTE ---
Test costruttore e funzione
aggiungi
- Caricatore
- Cuffie
- Macchina fotografica
- Documenti
- Biglietti aereo
- Vestiti

Test funzione prendi
X Caricatore
- Cuffie
X Macchina fotografica
X Documenti
- Biglietti aereo
- Vestiti

Test funzione viaggia
- Caricatore
- Cuffie
- Macchina fotografica
- Documenti
- Biglietti aereo
- Vestiti

Test costruttore di copia
- Caricatore
- Cuffie
- Macchina fotografica
- Documenti
- Biglietti aereo
- Vestiti
- Oggetto 1
- Oggetto 2
- Oggetto 3
```

```
--- SECONDA PARTE ---
Test eventuale distruttore
Distruttore chiamato
Test operatore +=
- Caricatore
X Cuffie
X Macchina fotografica
- Documenti
- Biglietti aereo
- Vestiti
- Spazzolino
- Dentifricio
X Occhiali da sole

Test funzione rimuovi
- Documenti
- Biglietti aereo
- Vestiti
- Spazzolino
- Dentifricio
X Occhiali da sole

Test operatore di negazione logica
- Biglietti aereo
- Spazzolino
- Dentifricio
```

Note per la consegna:

Affinché l'elaborato venga considerato valido, il programma **deve** produrre almeno la prima parte dell'output atteso. In questo caso, i docenti procederanno alla valutazione dell'elaborato **solo se** lo studente avrà completato l'autocorrezione del proprio elaborato.

In **tutti** gli altri casi (per esempio, il programma non compila, non collega, non esegue o la prima parte dell'output non coincide con quella attesa), l'elaborato è considerato **insufficiente** e, pertanto, **non verrà corretto**.