Un `UfficioPostale` è costituito da `M` sportelli a cui gli utenti possono accodarsi per essere serviti. Gli sportelli sono numerati a partire da 1. Ogni coda può contenere un numero illimitato di utenti. Ogni utente è caratterizzato da: (i) un nome, che è una stringa non vuota di al più 25 caratteri; e (ii) da un attributo, che specifica se tale utente è o non è un utente prioritario.

Implementare le seguenti operazioni che possono essere effettuate su `UfficioPostale`:

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

- ✓ **`UfficioPostale up(M);`**

Costruttore che inizializza un `UfficioPostale` costituito da `M` sportelli. Di default `M` vale 2, valore da utilizzare anche in caso di necessità di sanitizzazione. `UfficioPostale` è inizialmente vuoto, cioè non vi sono utenti in coda agli sportelli.

- ✓ **`cout << up;`**

Operatore di uscita per il tipo `UfficioPostale`, che stampa a schermo lo stato di `up` nel seguente formato:

```
Utenti totali: 4
Prioritari: 2
- Sportello 1: Maria Rossi (P), Luigi Bianchi, Daniela Neri
- Sportello 2: Giuseppe Verdi (P)
```

Come si nota dall'esempio, viene innanzitutto stampato il numero totale di utenti in `up` ed il numero di utenti prioritari. Gli sportelli sono stampati in ordine crescente per numero di sportello. La lista di utenti in coda ad uno sportello è stampata in ordine di fila. Nell'esempio di cui sopra, Maria Rossi è il prossimo utente ad essere servito allo sportello 1, seguito da Luigi Bianchi e Daniela Neri. Per ogni utente, oltre ad essere stampato il nome, viene anche stampata un'indicazione se tale utente è prioritario. In particolare, in caso di utente prioritario, il nome viene seguito da "(P)", come si può vedere nel caso di Maria Rossi e Giuseppe Verdi. Per gli utenti non prioritari viene stampato solo il nome. I nomi degli utenti sono separati da una virgola ed uno spazio. Si noti inoltre la presenza di un singolo spazio anche dopo ciascun carattere ':'.

- ✓ **`up.accodaUtente(n, sp);`**

Metodo che aggiunge ad `UfficioPostale up` un nuovo utente non prioritario caratterizzato dal nome `n`. L'utente viene aggiunto in fondo alla coda relativa allo sportello con numero `sp`. Se `n` è già presente nella coda relativa allo sportello `sp`, l'utente non viene aggiunto. In caso di input non validi, il metodo fallisce e `up` rimane inalterato.

- ✓ **`up.serviUtente(sp);`**

Questo metodo modifica `up` servendo il primo utente in coda allo sportello avente numero `sp`. Servire un utente vuol dire rimuovere tale utente dalla coda.

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

- ✓ **`~UfficioPostale();`**

Qualora sia necessario, implementare il distruttore.

- ✓ **`up.accodaPrioritario(n);`**

Metodo che modifica `up`, accodando un utente prioritario contraddistinto dal nome `n`. Tra tutti gli sportelli, tale utente viene aggiunto alla coda contenente il minor numero di utenti prioritari. Se due code hanno lo stesso numero minimo di utenti prioritari, il nuovo utente prioritario viene aggiunto alla coda relativa al numero di sportello minore. Se un utente con stesso nome `n` è già presente in coda o se `n` non rientra in 25 caratteri, il nuovo utente prioritario non viene aggiunto e `up` rimane invariato. Altrimenti, il nuovo utente prioritario viene accodato davanti ad eventuali utenti non prioritari nella stessa coda e dietro eventuali altri utenti prioritari precedentemente accodati. Considerando l'esempio relativo all'operatore di stampa (si veda la pagina precedente), l'utente prioritario Simona Gialli verrebbe accodato allo sportello 1 dopo Maria Rossi e prima di Luigi Bianchi.

- ✓ **`up.passaAvanti(n, sp, pos);`**

Metodo che modifica `up`, facendo avanzare di un numero di posizioni `pos` l'utente caratterizzato da nome `n` e accodato allo sportello avente numero `sp`. Qualora tale utente dovesse essere non prioritario e, avanzando, superasse almeno un utente prioritario, diventerebbe prioritario a sua volta. Nell'esempio relativo all'operatore di stampa, se Luigi Bianchi dovesse avanzare di una posizione diventerebbe un utente prioritario. Il metodo fallisce in caso di input non validi oppure nel caso in cui l'utente avente nome `n` non fosse presente nella coda allo sportello `sp`. Si noti che `pos` non può superare il numero di utenti presenti in coda in quello sportello: qualora lo fosse, il metodo lascerebbe l'oggetto up invariato.

- ✓ **`!up;`**

Operatore di negazione logica di un `UfficioPostale`, che modifica `up` invertendo la priorità degli utenti: tutti gli utenti prioritari diventano non prioritari, e viceversa. La coda ad ogni sportello deve essere aggiornata, in modo che gli utenti diventati prioritari siano in testa alla coda e gli utenti diventati non prioritari vadano in fondo. Non deve essere modificato l'ordine degli utenti appartenenti alla stessa categoria. Nell'esempio relativo all'operatore di stampa, Luigi Bianchi viene prima di Daniela Neri anche dopo aver eseguito questo operatore. Questo operatore restituisce un riferimento ad `up`.

Mediante il linguaggio C++, realizzare il tipo di dato astratto **`UfficioPostale`**, 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 accodaUtente

Utenti totali: 4
Prioritari: 0
- Sportello 1: Maria Rossi, Luigi Bianchi, Chiara Biondi
- Sportello 2: Giuseppe Verdi
- Sportello 3:


Test funzione serviUtente

Utenti totali: 3
Prioritari: 0
- Sportello 1: Luigi Bianchi, Chiara Biondi
- Sportello 2: Giuseppe Verdi
- Sportello 3:


--- SECONDA PARTE ---
Test eventuale distruttore
Distruttore chiamato


Test funzione accodaPrioritario

Utenti totali: 7
Prioritari: 4
- Sportello 1: Simona Gialli (P), Maurizio Violi (P), Luigi Bianchi, Chiara Biondi
- Sportello 2: Daniela Neri (P), Giuseppe Verdi
- Sportello 3: Francesco Grigi (P)


Test funzione passaAvanti

Utenti totali: 7
Prioritari: 5
- Sportello 1: Simona Gialli (P), Maurizio Violi (P), Chiara Biondi, Luigi Bianchi
- Sportello 2: Giuseppe Verdi (P), Daniela Neri (P)
- Sportello 3: Francesco Grigi (P)


Test operatore di negazione

Utenti totali: 7
Prioritari: 2
- Sportello 1: Chiara Biondi (P), Luigi Bianchi (P), Simona Gialli, Maurizio Violi
- Sportello 2: Giuseppe Verdi, Daniela Neri
- Sportello 3: Francesco Grigi
```

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**.