Un `GestoreMessaggi` rappresenta un'applicazione per lo scambio di messaggi tra utenti registrati. Ogni utente è rappresentato da un nickname univoco di almeno un carattere e di al più 20. Per ogni utente, il `GestoreMessaggi` mette a disposizione una casella postale in grado di contenere fino a 5 messaggi non letti. Per ogni messaggio, la casella mette a disposizione al più 30 byte. Implementare le seguenti operazioni che possono essere effettuate su un `GestoreMesaggi`:

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

- ✓ **`GestoreMessaggi g(n);`**

Costruttore che inizializza un `GestoreMessaggi` capace di contenere `n` utenti. Inizialmente `g` non contiene alcun utente e tutte le caselle postali sono vuote. Il numero minimo di utenti per un `GestoreMessaggi` è 2, sanitizzare a tale valore quando necessario.

- ✓ **`g.registra_utente(id);`**

Funzione che registra l'utente con nickname `id` presso `g`. Rimuovere eventuali spazi bianchi ad inizio e fine nickname prima di valutarne la correttezza. In caso di duplicato, input scorretto o `GestoreMessaggi` al completo la funzione restituisce `false`, altrimenti `true`. Un utente registrato non potrà più essere eliminato.

- ✓ **`g.invia_messaggio(dest, mitt, testo);`**

Funzione che invia il messaggio `testo` al destinatario `dest` con mittente `mitt` (analogamente a prima, nei nickname vengono ignorati gli spazi iniziali e finali). Qualora il messaggio sia vuoto, troppo lungo per la casella di testo, almeno uno dei due utenti non sia registrato o la casella postale del destinatario sia piena, la struttura dati rimane inalterata.

- ✓ **`cout << g;`**

Operatore di uscita per il tipo `GestoreMessaggi`. L'output deve avere la seguente formattazione:

```
Numero utenti registrati: 4
Numero spazi disponibili: 6

Carlo: 5 messaggi da leggere
Sara: 2 messaggi da leggere
nick_3: 0 messaggi da leggere
nick_4: 1 messaggi da leggere
```

dove "Numero spazi disponibili" rappresenta il numero di registrazioni che `g` può ancora accettare. I nickname sono ordinati per ordine di iscrizione.

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

✓ **`~ GestoreMessaggi();`**

Distruttore, *qualora necessario*.

✓ **`g.leggi_messaggio(dest, mitt);`**

Funzione che permette di leggere il primo messaggio disponibile destinato a `dest` (analogamente a prima, nei nickname vengono ignorati gli spazi iniziali e finali). Se `dest` non esiste oppure se non vi è alcun messaggio, la funzione lascia la struttura dati inalterata e restituisce `nullptr`. Altrimenti, il messaggio viene eliminato dalla casella postale in quanto letto, scrive in `mitt` (assumendo sia ben formato) il nickname del mittente e la funzione restituisce un puntatore al messaggio letto. Il lettore non deve essere in grado di alterare il messaggio del mittente.

✓ **`GestoreMessaggi g1(g);`**

Costruttore di copia.

✓ **`g1 = n + g;`**

Operatore di somma tra un intero ed un `GestoreMessaggi` che crea e restituisce un nuovo `GestoreMessaggi` identico a `g` ma con con una capienza di `n` utenti maggiore. Se `n` è negativo, considerarlo 0. L'ordine in cui gli utenti risultano registrati deve rimanere il medesimo.

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

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---

Test del costruttore:
Numero utenti registrati: 0
Numero spazi disponibili: 4


Test della registra_utente:
Numero utenti registrati: 1
Numero spazi disponibili: 3

Utente1: 0 messaggi da leggere


Numero utenti registrati: 2
Numero spazi disponibili: 2

Utente1: 0 messaggi da leggere
Utente2: 0 messaggi da leggere


Test della invia_messaggio:
Numero utenti registrati: 2
Numero spazi disponibili: 2

Utente1: 1 messaggi da leggere
Utente2: 0 messaggi da leggere


Numero utenti registrati: 2
Numero spazi disponibili: 2

Utente1: 1 messaggi da leggere
Utente2: 1 messaggi da leggere


Numero utenti registrati: 2
Numero spazi disponibili: 2

Utente1: 2 messaggi da leggere
Utente2: 1 messaggi da leggere
```

```
--- SECONDA PARTE ---

Test della leggi_messaggio:
Mess 1 di u2 per u1
Utente1


Test del costruttore di copia:
Numero utenti registrati: 2
Numero spazi disponibili: 2

Utente1: 2 messaggi da leggere
Utente2: 0 messaggi da leggere


Test dell'operatore di somma:
Numero utenti registrati: 2
Numero spazi disponibili: 5

Utente1: 2 messaggi da leggere
Utente2: 0 messaggi da leggere
```

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**.