Un `ExchangeClient` rappresenta un client del servizio di scambio dati "Exchange". Esso permette di scambiare dati sotto forma di stringhe di lunghezza al più 20 caratteri con altri client definiti nello stesso programma. Lo scambio avviene in maniera asincrona attraverso l'utilizzo di buffer, ovvero un mittente scrive il dato da scambiare in un buffer ed il destinatario potrà leggerlo in un momento successivo. Lo scambio di dati può avvenire in due modi: unicast o broadcast. Nel primo caso il mittente specifica un solo destinatario, nel secondo i destinatari sono tutti i client definiti nel programma. Non possono esserci più di 5 client nello stesso programma (vedi costruttore di default). I client hanno a disposizione 4 buffer per le comunicazioni unicast ed 1 buffer per le comunicazioni broadcast. Ogni buffer può contenere al massimo un dato. All'inizio del programma tutti i buffer sono vuoti.

Implementare le seguenti operazioni che possono essere effettuate su un `ExchangeClient`.

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

- ✓ **`ExchangeClient e;`**

Costruttore di default che inizializza un `ExchangeClient`. Esso provvede ad assegnare al client un identificativo immutabile avente valore da 1 a 5, in maniera tale che al primo client creato venga assegnato l'identificativo 1, al secondo 2, ecc. Una volta assegnato un identificatore ad un client, esso non può essere mai più riassegnato. Il sistema non accetta più di 5 client, quindi se il programma tenta di creare un sesto client, verrà invocata la funzione `exit(1)` per terminare il programma.

- ✓ **`e.scrivi(dato, d);`**

Metodo che permette al client di inviare il dato `dato` al client destinatario `d`. Se il destinatario è un valore tra 1 e 5, esso è da intendersi come l'identificativo del destinatario (che non può coincidere con quello del mittente); se invece ha valore 0, il messaggio è da intendersi broadcast. Un messaggio unicast non può essere scritto se non è disponibile un buffer vuoto per i messaggi unicast. Se invece esistono buffer vuoti, il buffer da utilizzare per la scrittura è sempre il primo vuoto successivo all'ultimo scritto (in senso circolare). All'avvio del programma, quando cioè nessun buffer è ancora mai stato scritto, la prima scrittura userà il buffer 1. Analogamente, un messaggio broadcast non può essere scritto se il buffer per i messaggi broadcast non è vuoto. I destinatari di un messaggio broadcast sono tutti e soli i client definiti al momento dell'invio del messaggio. Se il `dato` è troppo lungo per stare in un buffer, il metodo provvede a troncarlo. La scriittura non fallisce in caso l'identificativo del destinatario non sia ancora assegnato. In caso di scrittura avvenuta correttamente, il metodo restituisce `true`, altrimenti la struttura dati rimane inalterata ed il metodo restituisce `false`.

- ✓ **`cout << e;`**

Operatore di uscita per il tipo `ExchangeClient`. Esso provvede a fornire una overview dello stato del servizio a fini di diagnostica. Il formato è il seguente:

```
B,mitt:1,dato:"dato 1"
1,mitt:3,dest:1,dato:"dato 2"
2,mitt:5,dest:2,dato:"dato 6"
3,mitt:1,dest:3,dato:"dato 3"
4,vuoto
```

Dove la riga che inizia con "B" descrive il contenuto del buffer per le comunicazioni broadcast, quella che inizia con "1" descrive il contenuto del primo buffer, e così via. In particolare, il mittente del messaggio in broadcast è quello identificato dal valore "1" ed il dato condiviso è la stringa "dato 1", il mittente del

messaggio contenuto nel buffer 1 è il client "3", ha come destinatario il client "1" e contiene il dato "dato 2", ecc. Il buffer 4 è vuoto.

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

- ✓ **`~ExchangeClient();`**

Qualora sia necessario, implementare il distruttore.

- ✓ **`e.leggi(dato, m);`**

Metodo che permette al client `e` di leggere un messaggio a sé indirizzato e proveniente dal mittente `m` scrivendolo in `dato`, considerato già allocato dal chiamante. Il dato da leggere deve essere sempre quello nel buffer con indice minore, salvo in caso ve ne sia uno in broadcast, il quale ha sempre priorità. Il mittente è sempre indicato con un numero da 1 a 5. Se non vi sono messaggi per tale client, il metodo restituisce `false` e non modifica il contenuto di `dato`, altrimenti `true`. Se il dato è stato letto da tutti i suoi destinatari, il metodo svuota il buffer.

- ✓ **`(int)e;`**

Operatore di conversione ad intero di un `ExchangeClient` che restituisce il numero di messaggi aventi tra i destinatari il proprietario di `e`.

- ✓ **`ExchangeClient::cifra(dato, shift_c, shift_p);`**

Metodo che modifica il dato `dato` cifrandolo come segue. i) Tutti i caratteri alfanumerici vengono sostituiti con lo `shift_c`-esimo carattere successivo *nella propria classe di appartenenza*. Tali classi sono: i numeri (0, 1, 2, …), le lettere minuscole (a, b, c, …), e le lettere maiuscole (A, B, C, …). In caso di "overflow" il metodo esegue lo shift in maniera circolare nella sua classe di appartenenza. Se il carattere non è alfanumerico viene lasciato inalterato. ii) Tutti i caratteri vengono spostati nella posizione `shift_p`-esima precedente. Anche in questo caso il metodo gestisce i casi di overflow in maniera circolare. Sia `shift_c` che `shift_p` possono essere zero o negativi.

Mediante il linguaggio C++, realizzare il tipo di dato astratto **`ExchangeClient`**, definito dalle precedenti specifiche. Non è permesso l'utilizzo di variabili globali, né delle funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc. **Gestire le eventuali situazioni di errore**.

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**.

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---

Test del costruttore ed operatore di uscita:
B,vuoto
1,vuoto
2,vuoto
3,vuoto
4,vuoto

Test della scrivi
1 1 1 1 1 0

B,mitt:1,dato:"dato 2"
1,mitt:1,dest:2,dato:"dato 1"
2,mitt:2,dest:1,dato:"dato 3"
3,mitt:2,dest:1,dato:"dato 4"
4,mitt:3,dest:1,dato:"dato 5"

--- SECONDA PARTE ---

Test operatore di conversione ad intero
3 2 1

Test della leggi
1
dato 3
1
dato 5
1
dato 2
1
dato 2

B,vuoto
1,mitt:1,dest:2,dato:"dato 1"
2,vuoto
3,mitt:2,dest:1,dato:"dato 4"
4,vuoto


Test della cifra
cXYABC89012xyzab
```