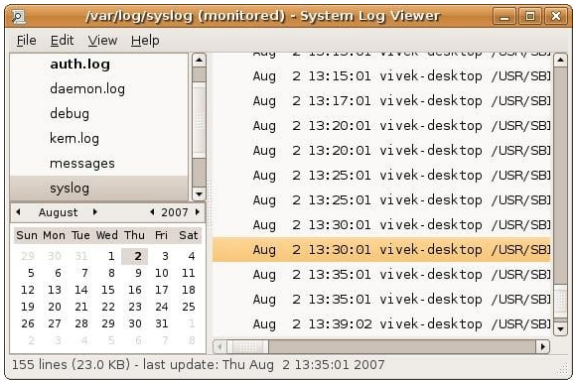

Un `Log` rappresenta una sequenza di eventi di log di un sistema operativo o un'applicazione. Ogni evento di log è rappresentato da una data e ora, da un livello di priorità, e da una descrizione di al più 80 caratteri. Il livello di priorità può assumere uno dei seguenti valori (in ordine di priorità crescente): informazione, avviso, errore, evento critico.

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

--- PRIMA PARTE ---

✓ **`Log l;`**

Costruttore di default che inizializza un `Log` vuoto, in cui cioè non ci sono eventi.

✓ **`cout << l;`**

Operatore di uscita per il tipo `Log`. L'uscita ha la forma seguente:

```
--- LOG ---
INFO,2023-03-15,17:33,Applicazione 1 avviata
INFO,2023-03-15,17:45,Applicazione 2 avviata
WARN,2023-03-15,18:01,Applicazione 2 non puo' accedere la risorsa A
INFO,2023-03-15,18:04,Applicazione 3 avviata
WARN,2023-03-15,18:07,Applicazione 1 ha tentato di aprire file inesistente B
INFO,2023-03-15,18:20,(cancellato)
INFO,2023-03-15,18:24,Applicazione 1 arrestata dall'amministratore
ERRO,2023-03-15,18:26,Applicazione 2 ha tentato di dividere per zero: arrestata
CRIT,2023-03-15,18:59,Buffer overflow nell'applicazione 3! Allarme di sicurezza!
INFO,2023-03-15,19:01,Applicazione 3 arrestata dall'amministratore
--- FINE LOG ---
```

Notare che gli eventi di log sono stampati in ordine cronologico, dal più vecchio al più recente. Nel caso due eventi abbiano la stessa data e ora, sono stampati nell'ordine in cui sono stati registrati (vedi funzione `registra`). Tutto il `Log` è preceduto da una riga "**`--- LOG ---`**" e terminato da una riga da una riga "**`--- FINE LOG ---`**". Ogni riga riporta un singolo evento, con un formato del tipo "**`[Priorità],[Data],[Ora],[Messaggio]`**". La priorità è espressa da quattro lettere maiuscole: "**`INFO`**" (informazione), "**`WARN`**" (avviso), "**`ERRO`**" (errore) e "**`CRIT`**" (evento critico). La data è nel formato "**`AAAA-MM-GG`**", ed ha una lunghezza fissa di 10 caratteri. L'ora è nel formato "**`OO:MM`**", ed ha una lunghezza fissa di 5 caratteri. Gli eventi cancellati (vedi funzione `cancella`) sono espressi con il messaggio "`(cancellato)`".

✓ **`l.registra(prio, data, ora, msg);`**

Operazione che registra un nuovo evento nel `Log` `l`, con priorità `prio`, data `data`, ora `ora`, e un messaggio non vuoto `messaggio`. L'argomento `prio` vale '`i`' nel caso di informazioni, '`w`' nel caso di avvisi, '`e`' nel caso di errori, '`c`' nel caso di eventi critici. Data e ora sono strutture ad-hoc da dichiarare insieme alla classe. La struttura per la data contiene tre membri interi: `anno` (da 1970 a 9999), `mese` (da 1 a 12), `giorno` (da 1 a 30). Supponiamo per semplicità che non siano ammessi anni precedenti al 1970 né successivi al 9999, e che tutti i mesi abbiamo 30 giorni. La struttura per l'ora contiene due membri interi: `ore` (da 0 a 23), `minuti` (da 0 a 59). Se gli input non sono del formato giusto, il `Log` rimane inalterato.

✓ **`l.cancella(msg);`**

Operazione che cancella un messaggio da un `Log` l'evento non ancora cancellato con messaggio `msg`. Nel caso due eventi non cancellati abbiano lo stesso messaggio, viene cancellato quello con data e ora più vecchie. Nel caso abbiano anche le stesse data e ora, viene cancellato il primo ad essere stato registrato. Un evento cancellato non viene rimosso totalmente dal `Log`, ma viene

rimosso solo il suo messaggio. Se un evento non ancora cancellato con quel messaggio non esiste, o se l'input non è del formato giusto, il `Log` rimane inalterato.

--- SECONDA PARTE ---

✓ **`~Log();`**
Qualora sia necessario, implementare il distruttore.

✓ **`l.cancella(da_data, da_ora, a_data, a_ora);`**
Operazione che cancella dal `Log` tutti gli eventi da data `da_data` ora `da_ora` fino a data `a_data` ora `a_ora` comprese. Se gli input non sono validi, il `Log` rimane inalterato.

✓ **`l.filtra(prio);`**

Operazione che alloca e restituisce un puntatore ad un nuovo `Log` contenente tutti e soli gli eventi non cancellati di `l` che hanno priorità `prio`. L'argomento `prio` assume valore come nella funzione `registra`. Se gli input non sono validi, viene restituito `nullptr`.

Nota: un'implementazione ottimizzata senza cicli innestati verrà valutata maggiormente.

✓ **`l.biforca(data, ora, m);`**
Operazione che alloca e restituisce un puntatore ad un nuovo `Log`, risultato di una biforcazione di `l`. Il `Log` risultato è identico a `log1` fino alla data `data` ora `ora` comprese, dopo le quali è identico al `Log` `m`. Per esempio, se dal `Log` di cui sopra si calcola una biforcazione in data `2023-03-15` e ora `18:26` e `m` uguale a:

```
--- LOG ---
ERRO,2023-03-15,18:07,L'applicazione 3 non puo' aprire la porta 80
ERRO,2023-03-15,18:26,L'applicazione 3 non puo' aprire nemmeno la porta 443
WARN,2023-03-15,18:41,File C non disponibile nella cartella D
ERRO,2023-03-15,19:11,(cancellato)
CRIT,2023-03-15,19:25,OS arrabbiato perche' errore precedente cancellato
--- FINE LOG ---
```

allora il `Log` risultante sarà:

```
--- LOG ---
INFO,2023-03-15,17:33,Applicazione 1 avviata
INFO,2023-03-15,17:45,Applicazione 2 avviata
WARN,2023-03-15,18:01,Applicazione 2 non puo' accedere la risorsa A
INFO,2023-03-15,18:04,Applicazione 3 avviata
WARN,2023-03-15,18:07,Applicazione 1 ha tentato di aprire file inesistente B
INFO,2023-03-15,18:20,(cancellato)
INFO,2023-03-15,18:24,Applicazione 2 arrestata dall'amministratore
ERRO,2023-03-15,18:26,Applicazione 2 ha tentato di dividere per zero: arrestata
WARN,2023-03-15,18:41,File C non disponibile nella cartella D
ERRO,2023-03-15,19:11,(cancellato)
CRIT,2023-03-15,19:25,OS arrabbiato perche' errore precedente cancellato
--- FINE LOG ---
```

Se gli input non sono validi, viene restituito `nullptr`.

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

USCITA ATTESA

```
--- PRIMA PARTE ---
Test del costruttore:
--- LOG ---
--- FINE LOG ---

Test della registra:
--- LOG ---
INFO,2023-03-15,17:33,Applicazione 1 avviata
INFO,2023-03-15,17:45,Applicazione 2 avviata
WARN,2023-03-15,18:01,Applicazione 2 non puo' accedere la risorsa A
INFO,2023-03-15,18:04,Applicazione 3 avviata
WARN,2023-03-15,18:07,Applicazione 1 ha tentato di aprire file inesistente B
INFO,2023-03-15,18:20,Evento che verra' cancellato
INFO,2023-03-15,18:24,Applicazione 1 arrestata dall'amministratore
ERRO,2023-03-15,18:26,Applicazione 2 ha tentato di dividere per zero: arrestata
CRIT,2023-03-15,18:59,Buffer overflow nell'applicazione 3! Allarme di sicurezza!
INFO,2023-03-15,19:01,Applicazione 3 arrestata dall'amministratore
--- FINE LOG ---

Test della cancella:
--- LOG ---
INFO,2023-03-15,17:33,Applicazione 1 avviata
INFO,2023-03-15,17:45,Applicazione 2 avviata
WARN,2023-03-15,18:01,Applicazione 2 non puo' accedere la risorsa A
INFO,2023-03-15,18:04,Applicazione 3 avviata
WARN,2023-03-15,18:07,Applicazione 1 ha tentato di aprire file inesistente B
INFO,2023-03-15,18:20,(cancellato)
INFO,2023-03-15,18:24,Applicazione 1 arrestata dall'amministratore
ERRO,2023-03-15,18:26,Applicazione 2 ha tentato di dividere per zero: arrestata
CRIT,2023-03-15,18:59,Buffer overflow nell'applicazione 3! Allarme di sicurezza!
INFO,2023-03-15,19:01,Applicazione 3 arrestata dall'amministratore
--- FINE LOG ---
```

```
--- SECONDA PARTE ---
Test dell'(eventuale) distruttore:
(distruttore invocato)
Test della cancella overloaded:
--- LOG ---
INFO,2023-03-15,17:33,Applicazione 1 avviata
INFO,2023-03-15,17:45,Applicazione 2 avviata
WARN,2023-03-15,18:01,Applicazione 2 non puo' accedere la risorsa A
INFO,2023-03-15,18:04,(cancellato)
WARN,2023-03-15,18:07,(cancellato)
INFO,2023-03-15,18:20,(cancellato)
INFO,2023-03-15,18:24,(cancellato)
ERRO,2023-03-15,18:26,Applicazione 2 ha tentato di dividere per zero: arrestata
CRIT,2023-03-15,18:59,Buffer overflow nell'applicazione 3! Allarme di sicurezza!
INFO,2023-03-15,19:01,Applicazione 3 arrestata dall'amministratore
--- FINE LOG ---

Test della filtra:
--- LOG ---
INFO,2023-03-15,17:33,Applicazione 1 avviata
INFO,2023-03-15,17:45,Applicazione 2 avviata
INFO,2023-03-15,19:01,Applicazione 3 arrestata dall'amministratore
--- FINE LOG ---

Test della biforca:
--- LOG ---
INFO,2023-03-15,17:33,Applicazione 1 avviata
INFO,2023-03-15,17:45,Applicazione 2 avviata
WARN,2023-03-15,18:01,Applicazione 2 non puo' accedere la risorsa A
INFO,2023-03-15,18:04,(cancellato)
WARN,2023-03-15,18:07,(cancellato)
INFO,2023-03-15,18:20,(cancellato)
INFO,2023-03-15,18:24,(cancellato)
ERRO,2023-03-15,18:26,Applicazione 2 ha tentato di dividere per zero: arrestata
WARN,2023-03-15,18:41,File C non disponibile nella cartella D
ERRO,2023-03-15,19:11,(cancellato)
CRIT,2023-03-15,19:25,OS arrabbiato perche' errore precedente cancellato
--- FINE LOG ---
```

Nota finale. 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 (sia della prima che della seconda parte).

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