Una `History` rappresenta un insieme di eventi storici. Ogni evento storico è rappresentato da un anno e da una descrizione di al più 30 caratteri.

Implementare le seguenti operazioni che possono essere effettuate su una `History`.

--- PRIMA PARTE ---

✓ `History hist;`
Costruttore di default che inizializza una `History` vuota, in cui cioè non ci sono eventi.

✓ `cout << hist;`
Operatore di uscita per il tipo `History`. L'uscita ha la forma seguente:

```
-- HISTORY --
480 BC
Battle of Thermopylae
-----
460 BC
First Peloponnesian War
-----
218 BC
Second Punic War
-----
44 BC
Julius Caesar is assassinated
-----
64 AD
Much of Rome burns
-----
410 AD
Visigoths sack Rome
-----
476 AD
Last Roman emperor is deposed
-----
```

Notare che gli eventi sono stampati in ordine cronologico, dal più antico al più recente. Nel caso due eventi cadano nello stesso anno, sono stampati nell'ordine in cui sono stati registrati (vedi funzione `record`). Le date prima di Cristo sono seguite da "**BC**", mentre quelle dopo Cristo da "**AD**". Sotto ogni data è stampata la descrizione, e sotto la descrizione una riga di separazione "**-----**". Tutta la `History` è preceduta da una riga "**-- HISTORY --**".

✓ `hist.record(year, descr);`
Operazione che registra un nuovo evento storico su una `History`, accaduto nell'anno `year` e descritto da `descr`. Si noti che per convenzione storiografica, l'anno 0 non esiste e l'anno successivo al 1 BC è il 1 AD. Per semplicità, l'argomento `year` segue lo standard ISO 8601, che fa corrispondere gli anni prima di Cristo a numeri negativi o zero (`year=0` corrisponde al 1 BC, `year=-1` al 2 BC, e così via) e gli anni dopo Cristo a numeri positivi (`year=1` corrisponde al 1 AD, e così via). Se gli input non sono del formato giusto, la `History` rimane inalterata.

✓ `hist.forget(descr);`
Operazione che fa dimenticare ad una `History` l'evento storico descritto da `descr`. Nel caso due eventi abbiano la stessa descrizione, viene dimenticato quello più antico. Nel caso abbiano anche lo stesso anno, viene dimenticato il primo ad essere stato registrato. Se l'input non è del formato giusto, la `History` rimane inalterata.

✓ `~History();`
Distruttore.

--- SECONDA PARTE ---

✓ `hist.longest_period();`
Operazione che restituisce l'intervallo di tempo più lungo tra due eventi storici consecutivi in una `History`, misurato in anni. Tra due eventi accaduti rispettivamente il 5 BC ed il 20 AD passano 24 anni. Se una `History` non ha almeno due eventi storici, non è possibile calcolare intervalli di tempo, quindi la funzione `longest_period` restituirà il valore speciale -1.

✓ `hist.forget(from_year, to_year);`
Operazione che fa dimenticare alla `History` tutti gli eventi storici accaduti dall'anno `from_year` all'anno `to_year` compresi. Gli argomenti `from_year` e `to_year` seguono lo standard ISO 8601 sopra descritto. Se gli input non sono validi, la `History` rimane inalterata.

✓ `create_alternative(hist1, fork_year, hist2);`
Funzione globale che alloca e restituisce un puntatore ad una nuova istanza di `History`, alternativa a `hist1`. La `History` alternativa è identica a `hist1` fino all'anno di biforcazione `fork_year` compreso, dopo il quale è identica alla `hist2`. Per esempio, se dalla `History` di cui sopra si calcola un'alternativa con anno di biforcazione 300 AD e `hist2` uguale a:

```
-- HISTORY –
100 AD
Irrelevant event before fork
-----
410 AD
Visigoths fail to sack Rome
-----
1969 AD
First Roman astronaut on Moon
-----
```

allora la `History` alternativa sarà la seguente:

```
-- HISTORY --
480 BC
Battle of Thermopylae
-----
460 BC
First Peloponnesian War
-----
218 BC
Second Punic War
-----
44 BC
Julius Caesar is assassinated
-----
64 AD
Much of Rome burns
-----
410 AD
Visigoths fail to sack Rome
-----
1969 AD
First Roman astronaut on Moon
-----
```

L'argomento `fork_year` segue lo standard ISO 8601 sopra descritto.

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:
-- HISTORY --

Test della record:
-- HISTORY --
108 BC
ccc
-----
405 AD
ddd
-----
503 AD
aaa
-----
599 AD
bbb
-----
711 AD
eee
-----
902 AD
fff
-----

Test della forget:
-- HISTORY --
405 AD
ddd
-----
599 AD
bbb
-----
711 AD
eee
-----
902 AD
fff
-----

Test del distruttore:
(oggetto distrutto)
```

```
--- SECONDA PARTE ---
Test longest_period:
194
Test forget overloaded:
-- HISTORY --
405 AD
ddd
-----
902 AD
fff
-----

Test create_alternative:
-- HISTORY --
405 AD
ddd
-----
507 AD
hhh
-----
753 AD
iii
-----
821 AD
jjj
-----
```

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**.