Lookup database with barrier

Creare un programma lookup-database-with-barrier.c in linguaggio C che accetti invocazioni sulla riga di comando del tipo:

```
lookup-database-with-barrier <db-file-1> <db-file-2> <...> <db-file-n> <query-file-1> <query-file-2> <...> <query-file-n>
```

Il programma dovrà fondamentalmente leggere *n* file *database* con coppie del tipo *(nome, valore)* e ricercare al loro interno i nomi che compaiono nei file di query.
Ciascun file database è un file testuale in cui ogni riga ha la struttura *"nome:valore"* dove *nome* può contenere degli spazi e/o altri simboli e *valore* è un numero intero. I file di *query* sono file testuali con un nome per ogni riga. Per una maggiore comprensione, vedere i file d'esempio allegati.

Il programma, una volta avviato, creerà *n+1* thread che si coordineranno utilizzando una barriera ed eventuali strumenti di sincronizzazione aggiuntivi (a scelta dello studente).
I primi *n* thread, che denoteremo con DB-IN, si occuperanno di leggere i file *i*>, caricarli in memoria centrale e ricercare, al loro interno, le chiavi presenti nei file *i*> (non necessariamente presenti nel file db-file-*i*). L'ultimo thread, che denoteremo con C, si occuperà di memorizzare e visualizzare delle statistiche sulle query effettuate dai thread DB-IN.

Il programma dovrà prevedere almeno due strutture dati:

- shared_db: struttura dati condivisa tra i thread di tipo DB-IN; questa dovrà contenere:
  - una struttura dati dinamica, a scelta, implementata dallo studente; qui il prof usa una hash table
  - una barriera;
  - altri elementi ritenuti utili/necessari.
- shared_c: struttura dati condivisa tra i thread DB-IN e il thread C; questa dovrà contenere:
  - una terna *(thread_n, nome, valore)*
  - altri elementi ritenuti utili/necessari.

Il programma dovrà prevedere due fasi:

- Fase di caricamento: ciascun thread DB-IN-*i*, leggerà il file *i*> e ne caricarà il contenuto all'interno della struttura dati dinamica presente in shared_db; i thread dovranno coordinarsi per segnalare la fine di questa fase utilizzando una barriera;
- Fase di ricerca: ciascun thread DB-IN-*i* leggerà il file *i*> riga per riga e, di volta in volta, effettuerà una ricerca all'interno della struttura dati; se riscontrato un match (di tipo *case sensitive*), inserirà all'interno della struttura dati shared_c la terna *(thread_n, nome, valore)*, risvegliando opportunamente il thread C; seguono le specifiche della terna sopra citata:
  - *thread_n* rappresenta un identificatore del thread
  - *nome*: rappresenta la chiave cercata
  - *valore*: rappresenta il valore associato al *nome*

Il thread C manterrà in memoria un array di contatori di dimensione pari a n e che conterrà, in ciascuna posizione *i*, la somma dei valori ottenuti dalle ricerche effettuate dal thread DB-IN-*i*.

In tutte le fasi, C dovrà rimanere in attesa di nuove terne da "consumare" (in shared_c). Estratta una nuova terna, il thread C aggiornerà le statistiche presenti nell'array di contatori, incrementando il *thread_n*-esimo elemento dell'array.

Il programma, al termine dei lavori, dovrà terminare visualizzando le statistiche totali note al thread C e deallocando tutte le strutture dati utilizzate.

per questo la variabile done verrà impostata a zero ogni volta che estrae una terna di thread_n, nome valore