Esame 01

Descrizione del Problema

Vogliamo implementare un programma che gestisce un insieme di record. Ogni record è composto da un nome, un identificatore e da un valore intero. Non possono esistere due record con lo stesso identificatore; ma possono esistere due record con lo stesso nome o con lo stesso valore. Il programma deve permettere di cancellare un record esistente, di modificare il valore e il nome di un record esistente, di caricare i records da un file di testo e di salvare i records su un file di testo. Si può assumere che i nomi dei record siano stringhe di lunghezza massima 127.

Il formato dei files di testo è il seguente: ogni riga contiene un record e consiste di tre parti separate da spazi. La prima parte è l'identificatore del record, la seconda parte è il nome e la terza il valore intero. Per esempio, la riga `2716621 Pippo 5` contiene il record con indetificatore `2716621`, il nome `Pippo` e il valore `5`. Il file è composto da sole righe che contengono records, e non riporta a priori il numero dei records contenuti nel file.

La funziona di ricerca restituisce la posizione del record nell'array di records. Esiste una funzione di ricerca sia per cercare per identitificatore che per nome. Se il record non esiste, la funzione ritorna `-1`. Nel caso della ricerca per nome, la funzione ritorna la posizione del primo record con quel nome, e prende un valore aggiuntivo che indica la posizione da cui iniziare la ricerca.

La cancellazione viene fatta modificando un record esistente e assegnando all'identificatore il valore `-1`, al nome la stringa vuota, e al valore il numero `0`. Nella cancellazione non si devono spostare i record rimanenti. Il programma supporta anche la cancellazione di tutti i record con lo stesso nome.

L'inserimento di un nuovo record viene fatto nella prima posizione vuota, cioè con identificatore `-1`. Se non esiste una posizione vuota, si aggiunge un record alla fine dell'array.

Il programma permette anche di calcolare la somma dei valori dei record con lo stesso nome, e di contare i record validi.

Il problema va risolto implementando una serie di funzioni che operano su una struttura dati che rappresenta l'insieme dei records. La struttura dati deve essere implementata come un array dinamico. È consigliato usare un array dinamico che permette l'inserimento in coda in modo efficiente.

Suggerimento 1: È consigliato inizializzare il set di records con una dimensione iniziale di 4.

Suggerimento 2: Per la lettura di file è permesso leggere record fino a che non si incontra un errore di lettura, che si può testare con `if(fs)` o `while(fs)` dove `fs` è lo stream.

Suggerimento 3: Potete usare la funzione `print()`, già implementata, per stampare i record correnti.

Funzioni e Punteggio

- `RecordSet init()`: 2 punti
  - crea un array vuoto di records
- `void drop(RecordSet& rs)`: 1 punti
  - dealloca l'array di records
- `int insert(RecordSet& rs, int id, const char* name, int value)`: 6 punti

    - inserisce un nuovo record nella prima posizione libera o alla fine dell'array; inserisce il record solo se un record con lo stesso identificatore non esiste
    - ritorna la posizione del record inserito o `-1` se il record esiste già
- `int rcount(const RecordSet& rs)`: 1 punti
    - ritorna il numero dei record validi, cioè con identificatore diverso da `-1`
- `int vsum(const RecordSet& rs, const char* name)`: 2 punti
    - ritorna la somma dei valori dei record con nome dato
- `int search(const RecordSet& rs, int id)`: 2 punti
    - restituisce l'indice del record con identificatore dato o `-1` se il record non esiste
- `int find(const RecordSet& rs, const char* name, int start)`: 3 punti
    - restituisce l'indice del record con nome dato o `-1` se il record non esiste, iniziando la ricerca dalla posizione `start`
- `int update(RecordSet& rs, int id, int value)`: 2 punti
    - modifica il valore del record con identificatore dato
    - ritora la posizione del record cambiato o `-1` se il record non esiste
- `int remove(RecordSet& rs, int id)`: 2 punti
    - rimuove il record con nome dato
    - ritora la posizione del record eliminato o `-1` se il record non esiste
- `int erase(RecordSet& rs, const char* name)`: 3 punti
    - rimuove tutti i record con nome dato
    - ritora la posizione del primo record eliminato o `-1` se nessun record ha il nome dato
- `RecordSet load(const char* filename)`: 5 punti
    - carica i records dal file di testo
    - ritorna l'array di records caricati o l'array vuoto in caso di errore
- `int save(const RecordSet& rs, const char* filename)`: 4 punti
    - salva i records sul file di testo, escludendo i record cancellati
    - ritorna `-1` in caso di errore

Regole e Test

Il programma deve essere scritto nel file `esame01.cpp`. Potete madoficare il `main()` di quel file e usarlo come volete per testare le funzioni che implementate. Il programma compila e viene lanciato con il comando:

```
g++ esame01.cpp -o esame01 && ./esame01
```

Il programma viene testato automaticamente con il comando:

```
g++ test01.cpp -o test01 && ./test01
```

Il programma di test collauda le funzioni che avete implementato e assegna un voto a seconda di quanti test passano. Il voto è la somma dei punteggi elencati sopra e dipende solo dalla correttza con cui si passano i test. Si può consegnare solo se il tester ritorna almeno 18. Il voto massimo è 33 che corrisponde a 30 e lode.

Il test deve compilare e deve completare la sua esecuzione. In ogni altra evenienza il voto è 0.