Prova di laboratorio di Programmazione II del 14/09/18

NOME: COGNOME: MATRICOLA:

Si vuole creare un piccolo sistema per la gestione e l'accesso ad inverted index. Un *inverted index* è un particolare tipo di dizionario che viene utilizzato per accelerare la ricerca di parole in documenti. L'inverted index viene ad esempio usato da Google per individuare l'elenco delle pagine che contengono (parte delle) keyword specificate dall'utente o dalla funzione di Ricerca del sistema operativo. Un inverted index è rappresentato da un insieme di parole (le parole contenute nei documenti). Ad ogni parola è associato il numero totale di documenti in cui è presente la parola e una lista, detta *posting list*, degli identificatori dei documenti (numeri interi) in cui è presente la parola.
Nella cartella "14-09-18" sono presenti i file lista.h, lista.c, tipo.h, tipo.c per la gestione delle posting list, parola.h che contiene il tipo di dato `parola`, *inverted* che contiene i dati relativi ad un inverted index (prima riga: numero di parole, a seguire su ogni riga una parola, il numero totale dei documenti in cui è presente la parola, la sequenza di identificativi di documenti), *doc* che contiene i dati relativi ad un documento (id del documento nella prima riga, sequenza di parole contenute nel documento).

Testo problema	**Fatto**	**Val.**	**Max**
Punto 1: Creare un progetto (e il corrispondente makefile) per il caricamento dell'inverted index. Il progetto include il file *compito.cc* oltre ai moduli per l'implementazione dell'inverted index. Il file *compito.cc* deve contenere: • la funzione `parola* load()` che carica l'inverted index dal file *inverted*. Le parole con le relative informazioni devono essere memorizzate in un vettore dinamico di tipo `parola` della dimensione corrispondente al numero di parole (prima riga del file) che viene restituito dalla funzione stessa. • La procedura `stampa(parola*, int)` che stampa il contenuto dell'inverted index (primo parametro vettore dinamico, secondo parametro dimensione del vettore). • Un `main` che richiama in sequenza le due funzioni.			20
Punto 2: Estendere il progetto con la funzione di aggiornamento: • Estendere il file *compito.cc* aggiungendo la procedura `update(parola* &II, char* fileName)` che aggiorna l'inverted index caricando il contenuto del documento contenuto nel file `fileName`. Ogni file caricato ha la stessa struttura del file *doc*. Il codice deve gestire il caso di aggiunta di una parola, di aggiunta di un id di documento alla posting list di una parola già presente nell'inverted index. • Estendere il `main` affinché aggiorni l'inverted index con il documento contenuto nel file *doc*. Il `main` deve chiedere all'utente il nome del file da caricare, richiamare la procedura `update` e richiamare la funzione `stampa` per stampare l'inverted index risultante			5
Punto 3.a: Estendere il file *compito.cc* con la stampa dei documenti che soddisfano una richiesta "word1 AND word2". A tale scopo: • aggiungere al file *compito.cc* la procedura `void AND(parola* II, char* W1, char* W2)` che stampa l'elenco dei documenti che contengono entrambe le parole. • Estendere il `main` affinché chieda all'utente le parole e stampi il risultato richiamando la procedura `AND`.			4
Punto 3.b: Estendere il file *compito.cc* con la stampa dell'elenco ordinato dei documenti che soddisfano anche parzialmente una richiesta "word1 word2 … wordN" ovvero che contengono una o più parole specificate nella rischiesta. A tale scopo:			3

Prova di laboratorio di Programmazione II del 14/09/18

NOME: COGNOME: MATRICOLA:

• aggiungere al file *compito.cc* la funzione `int* match(parola* II, char** WL, int n)` che dato l'inverted index, il vettore dinamico delle parole e il numero totale di parole contenuto nel vettore, restituisce il vettore dinamico degli id dei documenti che contengono almeno una delle parole specificate, ordinate in ordine decrescente per il numero totale di parole nella richiesta contenute nel documento (ovvero dal documento che contiene più parole al documento che ne contiene meno). • Estendere il `main` affinché chieda all'utente di inserire una richiesta come sequenza di parole separate da spazio, acquisisca le parole in un vettore dinamico e le passi assieme all'inverted index alla funzione `match` e stampi il risultato.			
Voto			32

NOTE
La valutazione del codice prodotto avviene al termine della prova e punto per punto (dipendenze tra i punti: 1-2-3.a, 1-2-3.b). Per ogni punto completato, è fondamentale che il codice compili e rispetti le specifiche descritte nel punto stesso.
È VIETATO l'uso di break e di variabili globali.
Si ricorda che è possibile consultare solo il materiale cartaceo/digitale del corso e i libri di testo consigliati.
Al termine della prova, creare un folder nella directory /tmp/esame/risultato etichettato con il numero di matricola. All'interno devono essere presenti tutti i file .h e .cc del progetto e il makefile. Per essere valutati, il makefile deve produrre un eseguibile funzionante.

Dati di prova:

Il **Punto 1** deve stampare l'inverted index contenuto nel file *inverted*.

Il **Punto 2** deve stampare:

```
computer
4 documenti
1 2 4 5
laptop
2 documenti
1 3
tower
4 documenti
1 2 3 5
voltage
1 documenti
5
```

Il **Punto 3.a** data la richiesta `computer AND tower` deve stampare:
`1 2 5`

Il **Punto 3.b** data la richiesta `computer voltage` deve stampare :

```
5 1 2 4
```

Infatti il documento 5 contiene tutte e due le parole specificate mentre i documenti 1, 2 e 4 contengono una delle parole specificate (computer)