Esercizio per candidatura a Software Developer

L'esercizio può essere svolto con qualsiasi linguaggio a te consono, puoi utilizzare tutte le librerie che vuoi o altri componenti a tua scelta; allo stesso modo l'interfaccia utente che decidi di adottare è a tua discrezione.

Dovrà essere possibile, per noi, **vedere il codice sorgente e compilare (o eseguire, per i linguaggi interpretati) la tua soluzione** sull'ambiente Ubuntu Linux 16.04.1 x86-64 (ovvero amd64), supponi di partire da un'installazione desktop; potrai decidere di usare pacchetti o software da qualsiasi sorgente (apt-get, installazioni manuali, ecc), purché tu ci spieghi come fare o, preferibilmente, automatizzi la procedura di installazione.

Per la soluzione dell'esercizio, ci aspettiamo di ricevere il codice sorgente più README che ci dica come vedere che cosa hai fatto, e che spieghi che cosa hai fatto e perché lo hai fatto.

Puoi inviare la tua soluzione per email a **p.ferracin@trenolab.com** oppure condividerla su GitHub o GitLab a **pidue**.

Come valuteremo la soluzione:

In ordine d'importanza decrescente, verranno valutate:

Correttezza della soluzione;
Aderenza a principi di buono sviluppo software;
Chiarezza del codice;
Accettabilità delle performance. **Non è un esercizio in stile Hackerrank o Google Code Jam.** Non è rilevante cercare di scrivere un algoritmo ultra-ottimizzato che, per esempio, abbia complessità O(n) quando è disponibile un'alternativa semplice O(n log n). Non ci sarà punteggio pieno in caso di algoritmi inutilmente complessi (es. scrivere un algoritmo O(n^3) quando esiste una versione O(n) disponibile con strutture dati standard).

In sostanza: cerca di scrivere del codice che definiresti ben fatto e che non ti vergogneresti di mostrare ad un amico o collega, ricordando che, riguardo alle valutazioni, la correttezza stravince rispetto alle performance.

Puoi anche decidere di fare delle assunzioni; scrivile nel README.

Precisiamo che potrebbe non esistere un'unica soluzione giusta (anche se possono esistere moltissime soluzioni sbagliate)

Il problema da risolvere

Il software che creerai dovrà leggere due file CSV contenenti orari ferroviari, il primo con il tempo di passaggio pianificato (quando vorrei che il treno passi per una certa stazione) e il secondo con il tempo di passaggio effettivo (quando il treno è davvero passato in una certa stazione). Si vuole calcolare il ritardo a destinazione (ultimo passaggio) di ciascun treno e visualizzarlo, espresso in secondi, per i cinque treni più ritardati.

Ad esempio se il tempo pianificato è 2015-11-05T08:15:30+01:00 e il tempo effettivo è 2015-11-05T08:15:48+01:00, il ritardo è pari a 18 secondi. I treni potrebbero arrivare anche prima del tempo pianificato, il tal caso il ritardo è negativo.

Il file contiene i dati relativi a più treni, e i record sono in ordine cronologico. La prima riga del file contiene le intestazioni di colonna.

Il formato dei due file di input è il medesimo, puoi fare riferimento al seguente esempio.

Numero treno	Nome stazione	Ora di passaggio
4C34_854#2	LIVST	**2015-11-05T08:15:30+01:00**
2L98_724	SHENFLD	**2015-11-05T08:15:44+01:00**
4C34_854#2	BOWJ	**2015-11-05T08:16:41+01:00**
2L98_724	SHENLEJ	**2015-11-05T08:16:43+01:00**

Il formato del file di output dovrà essere di questo tipo:

Numero treno	Destinazione	Ora pianificata	Ora effettiva	Ritardo
4C34_854#2	SHENFLD	**09:16:40+01:00**	**09:16:52+01:00**	**12**
2L98_724	LIVST	**09:17:41+01:00**	**09:17:37+01:00**	**-4**

Puoi supporre che i due files si chiamino rispettivamente **planned.csv** e **actual.csv** e risiedano nella directory corrente. Il separatore dei campi è la virgola, il separatore dei record è il ritorno a capo standard per il sistema operativo che stai usando.