Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica
A.A. 2018-19

Compito d'esame -- 19 febbraio 2019 -- Compito B

Istruzioni (leggere attentamente)

Nota importante: la mancata osservanza delle seguenti regole può comportare la perdita di informazioni necessarie alla valutazione dell'esame.

Registrazione dei dati dello studente: PRIMA DI INIZIARE, eseguite (con un doppio click sull'icona) il programma `RegistraStudente` che si trova nella cartella `Esame`. Inserite (separatamente) *Numero di Matricola*, *Cognome* e *Nome* seguendo le istruzioni che compaiono sul terminale, e confermate i dati che avete inserito. Il programma genera il file `studente.txt` che contiene Matricola, Cognome e Nome su tre righe separate (nell'ordine indicato). Il file `studente.txt` non deve essere modificato manualmente. Verificate che i dati nel file `studente.txt` siano corretti. In caso di errore potete rieseguire il programma `RegistraStudente`.

Svolgimento degli esercizi: Per ogni esercizio avete una cartella Eserc**N** che contiene un file dal nome `B_Ex`**`N`**`.py` (dove **`N`** è il numero dell'esercizio) con lo scheletro della soluzione. Questo file incorpora un codice di test che proverà la vostra soluzione per un certo numero di possibili dati in input. Aprite il file con Spyder e modificate SOLO il contenuto della funzione. Eseguendo il file `.py` si otterrà il responso dei test sulla console. Non spostate i file dalla loro posizione e non create nuovi file. **Si noti che per la correzione verranno usati insiemi di dati di test diversi.**

E' possibile consultare la documentazione ufficiale del linguaggio Python, ma **non è possibile usare libri o appunti**. In caso di problemi tecnici chiedere ai docenti o ai tecnici del laboratorio.

Al termine dello svolgimento del compito d'esame **NON è necessario effettuare una procedura di consegna** dell'elaborato. Le vostre soluzioni saranno copiate direttamente dalla cartella `Esame` della macchina su cui avete lavorato. Quindi, per concludere la prova, è sufficiente chiudere il programma Spyder e lasciare la vostra postazione.

Esercizi

- **B_Ex1(l)** Scrivere una funzione che prende in ingresso una lista **l** contenente stringhe e restituisce la lista ordinata in modo crescente e senza ripetizioni delle lettere estratte dagli elementi contenuti in **l** seguendo la seguente regola: se *s* è una stringa che occupa in **l** una posizione pari, allora si estraggono da *s* le lettere in posizione dispari, se *s* occupa una posizione dispari in **l**, si estraggono da *s* le lettere in posizione pari. Le posizioni si contano a partire dallo 0. Ad esempio, se `l=["mamma","sole","casa"]` la funzione deve restituire `["a","l","m","s"]`.

- **B_Ex2 (m)** Scrivere una funzione che prende in ingresso una matrice di numeri interi **m**, rappresentata come lista di liste, e conta quante posizioni (i,j) hanno la proprietà che la somma degli elementi della riga i è uguale alla somma degli elementi della colonna j. Ad esempio, se la matrice m vale:

1	**3**	4	5
2	3	**4**	2
1	7	3	3

  Allora la funzione deve restituire 2 poiché le posizioni (0,1) e (1,2) e nessun'altra, hanno questa proprietà.

- **B_Ex3** Scrivere una funzione che prende in ingresso un file di testo e restituisce un dizionario con chiave un carattere e valore la lista ORDINATA dei numeri di riga (contate a partire da 1) in cui quel carattere è quello meno frequente, considerando nell'analisi SOLO le vocali ("aeiou" sia maiuscole che minuscole).

Ad esempio, se il file contiene:

In Sicilia ad Aci Trezza,
un paesino nei pressi di Catania, vivono i Toscano,
soprannominati da tutti Malavoglia.

La funzione deve restituire il dizionario `{'I':[1],'A':[1],'e':[1],'u':[2, 3]}`. Lettere maiuscole e minuscole vanno considerate distinte. Notate che se un carattere non è mai tra i caratteri meno frequenti in una riga allora NON deve comparire nel dizionario e che per ogni riga vi possono essere più caratteri che compaiono il numero minimo di volte.

- **B_Ex4(file,n)** Scrivere una funzione che calcola se sia possibile volare dalla città City1 alla città City2 nello stesso giorno facendo un solo scalo. Questa funzione prende in ingresso un file csv **file** rappresentante tutti i voli in partenza da City1 e tutti quelli in arrivo a City2 nel seguente formato:

Città_di_partenza,Città_di_arrivo,Ora_partenza,Ora_arrivo

Potete assumere che:
- per tutti i voli o la Città_di_partenza sia City1 oppure la Città_di_arrivo sia City2,
- per fare scalo in una città bisogna che l'ora di arrivo sia STRETTAMENTE minore dell'ora di partenza del volo che porta a City2,
- le ore di partenza e di arrivo siano numeri interi compresi tra 0 e 23 (non dovete quindi preoccuparvi dei minuti),
- per un volo Ora_partenza sia sempre minore di Ora_arrivo,
- tutti i voli rientrino entro al massimo le 23 (cioè non ci sono voli che partono prima ed arrivano dopo mezzanotte).

La vostra funzione deve costruire la lista ORDINATA alfabeticamente di tutte le città in cui è possibile fare scalo per andare da City1 a City2. Ad esempio, se il file contiene:

Città_di_partenza,Città_di_arrivo,Ora_partenza,Ora_arrivo
City1,Lisbona,9,11
City1,Londra,7,9
Lisbona, City2,10,12
Londra, City2,12,14
City1,Parigi,8,9
Parigi, City2,17,18

allora la funzione deve restituire la lista `['Londra','Parigi']`, poiché è possibile andare da City1 a City2 in giornata facendo scalo a Londra o Parigi in modo che l'ora di arrivo del primo volo sia STRETTAMENTE minore dell'ora di partenza del secondo volo.