Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica A.A. 2021-22

Compito d'esame -- 14 febbraio 2022 -- Compito A

Istruzioni (leggere attentamente)

Nota importante: la mancata osservanza delle seguenti regole può comportare la perdita di informazioni necessarie alla valutazione della prova d'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`.

Tempo a disposizione: 1 ora e 30 minuti

Per risolvere gli esercizi in modo che possano essere successivamente corretti **è necessario scrivere la soluzione di ogni esercizio nel file .py relativo**, che trovate nella cartella dell'esercitazione (ad esempio, per l'esercizio 1 scrivete il vostro programma nel file Ex1.py, per l'esercizio 2, nel file Ex2.py, e così via). Notate che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. NON modificate questo codice, ma **SCRIVETE SOLO il contenuto della funzione**. 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.**

In ogni esercizio, se non diversamente richiesto, potete sempre assumere che gli input forniti siano coerenti con la traccia (ad esempio, se l'esercizio chiede di dare in input alla funzione una lista non vuota di stringhe, potete sempre assumere l'input sia in tale forma e non è necessario nel codice effettuare controlli per gestire casi diversi da questo, considerando, ad esempio, il caso di lista vuota).

Per gli esercizi relativi a lettura da file, la stringa in input che identifica il file è sempre comprensiva anche della sua estensione e il file risiede sempre nella stessa directory dell'esercizio.

Esercizi

- **Ex1(l,x,y)** Scrivere una funzione che riceve in ingresso una lista **l** contenente numeri interi e due numeri interi **x** e **y** e restituisce la più corta sottolista di **l** che inizia con una **x** e finisce con una **y**. Se c'è più di una sottolista più corta con queste caratteristiche, la funzione deve restituire la prima da sinistra a destra. Nel caso in cui non ci sia alcuna sottolista che inizia con una **x** e finisce con una **y** la funzione deve restituire una lista uguale ad **l**. Ad esempio, se **l**=`[1,2,2,0,3,1,3,4,2,1,7,3]`, **x**=2 e **y**=3, la funzione deve restituire `[2,0,3]`, mentre se **l**=`[4,5,2,3,6]`, **x**=2 e **y**=5, la funzione deve restituire una lista uguale ad **l**.

- **Ex2(m,s)** Scrivere una funzione che riceve in ingresso una matrice **m** rappresentata come lista di liste contenente caratteri ed una stringa **s**, e restituisce un dizionario le cui chiavi sono i caratteri di **m** che sono contenuti in **s** ma che compaiono in **s** solo in posizioni dispari, e per ogni chiave *k* il valore associato è un insieme contenente tutte le posizioni di *k* in **m**, dove per posizione si intende una coppia (tupla di dimensione due), avente come prima e seconda componente, rispettivamente, il numero di riga ed il numero di colonna di una occorrenza di *k* in **m**. Ad esempio, se **m**= `[['a','r','d'],['d','b','r']]` e

s= `'fredda'`, la funzione deve restituire `{'a':{(0,0)},'r':{(0,1),(1,2)}}` (notate che la `'b'` non compare in `'fredda'`, mentre la `'d'` compare in `'fredda'` sia in posizione pari che dispari e pertanto la `'b'` e la `'d'`non sono chiavi del dizionario restituito).

- **Ex3(file1,file2)** Scrivere una funzione che aiuti una software house a gestire le sue commesse in un mese e che prende in ingresso due file di testo (in formato csv), il file **file1** contiene le informazioni sul personale a disposizione, le sue disponibilità di tempo (nel mese di interesse), la sua competenza ed il suo costo orario. Il file csv **file1** ha il seguente formato:

  `NomeDipendente,OreDisponibili,Competenza`

  dove `NomeDipendente` è il nome univoco del dipendente, `OreDisponibili` è il numero di ore che il dipendente ha disponibili nel mese di interesse e `Competenza` è la competenza che il dipendente ha. Per semplicità, potete assumere che ci sia un solo dipendente che abbia una specifica competenza.

  Il file **file2** contiene le commesse che la software house ha ricevuto e che devono essere svolte nel mese di interesse ed ha il seguente formato:

  $\texttt{Committente,NumeroOre}_1\texttt{,Competenza}_1\texttt{,…, NumeroOre}_m\texttt{,Competenza}_m$

  Dove `Committente` è il nome ditta che richiede il lavoro, mentre per ogni valore di j tra 1 ed m, $\texttt{NumeroOre}_j$ indica il numero di ore necessarie di una persona che abbia la competenza richiesta ($\texttt{Competenza}_j$). Potete assumere che tutte le commesse abbiamo un diverso committente. Le commesse vengono analizzate nell'ordine in cui sono nel file **file2** e sono accettate solo se ci sono persone con le competenze richieste e tempo disponibile sufficiente per la commessa. La funzione deve calcolare (e restituire) un dizionario con chiave il nome della ditta committente e valore un altro dizionario con chiave la competenza richiesta e valore il nome del dipendente che ha quella competenza (e disponibilità di tempo). Se nessun dipendente ha quella competenza allora il valore deve essere `'Competenza Assente'`, se la competenza è presente ma manca la disponibilità di ore allora il valore deve essere `'Indisponibile'`. Ovviamente, per ogni commessa bisogna ridurre la disponibilità di ore del dipendente con quella competenza prima di analizzare la commessa precedente. Ad esempio, se il file **file1** contiene:

  ```
  Carlo,100,Java
  Giulia,35,DataBases
  Flavia,50,Python
  ```

  e il file **file2** contiene:

  ```
  Ditta1,20,Python,15,Java
  Ditta2,15,DataBases,10,MachineLearning
  Ditta3,10,Java,20,DataBases,15,Python
  Ditta4,20,Python
  Ditta5,10,Python
  ```

  allora la funzione deve restituire `{'Ditta1': {'Python': 'Flavia', 'Java': 'Carlo'}, 'Ditta2': {'DataBases': 'Giulia', 'MachineLearning': 'Competenza Assente'}, 'Ditta3': {'Java': 'Carlo', 'DataBases': 'Giulia', 'Python': 'Flavia'}, 'Ditta4': {'Python': 'Indisponibile'}, 'Ditta5': {'Python': 'Flavia'}}`. Notate che la commessa della `'Ditta2'` contiene una competenza che non si ha (`MachineLearning`) e quella della `'Ditta4'` richiede 20 ore di un esperto Python, ma alla persona competente in Python (`Flavia`) ne rimangono solo 15 (50 iniziali meno 20 per `'Ditta1'` e meno 15 per `'Ditta3'`).

  Nota: Potete assumere che le ore nei file **file1** e **file2** siano sempre convertibili in numeri interi.