Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica
A.A. 2019-20

Compito d'esame -- 16 settembre 2020 -- 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`.

Svolgimento degli esercizi: Per ogni esercizio avete una cartella Eserc**N** che contiene un file dal nome `A_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 IDLE 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.

Esercizi

- **A_Ex1(s)** Scrivere una funzione che riceve in ingresso una stringa **s** (che può essere anche vuota) contenente cifre (0123456789) ed il simbolo @. Quest'ultimo funge da separatore fra sottostringhe che rappresentano numeri interi positivi. Più precisamente, un numero è *contenuto* in **s** se la sottostringa che lo rappresenta:
  - contiene solo cifre e si trova fra due simboli @;
  - parte all'inizio di **s** ed è seguita da un @;
  - termina alla fine di **s** ed è preceduta da un @;
  - è l'unica sequenza di cifre contenuta in **s**.

  La funzione deve restituire la somma dei soli numeri pari contenuti in **s**. Se **s** è vuota, allora la funzione deve restituire 0. Se in s ci sono due (o più) simboli @ consecutivi, questi agiscono come se fossero un unico simbolo @. Ad esempio, se **s**=`'123@2@24@755'`, allora la funzione deve restituire `26`, se **s** =`'@23@44@244@22'`, allora la funzione deve restituire `310`, se **s**=`'22@@@2@'`, allora la funzione deve restituire 24, se s=`'12345678'`, allora la funzione deve restituire `12345678`.

- **A_Ex2(M)** Scrivere una funzione che prende in input una matrice **M** di dimensione nxm rappresentata come lista di liste, che contiene i voti ottenuti da m studenti universitari in n esami (i voti sono numeri interi positivi). Più precisamente, l'intero contenuto nella posizione i,j rappresenta il voto ottenuto dallo studente j per l'esame i. Se lo studente j non ha sostenuto l'esame i, il valore riportato nella posizione i,j è zero. La funzione deve restituire una lista di m numeri frazionari corrispondenti alla media dei voti di ciascuno studente, in modo che l'elemento in posizione k della lista in output corrisponda alla media dello studente k (cioè quello i cui voti sono riportati nella colonna k di **M**). Nel calcolo della media NON si devono considerare gli esami non sostenuti dallo studente (cioè quelli per i quali il

valore in **M** è 0). Inoltre, la media deve essere approssimata a due cifre decimali, usando la funzione `round()` (ad esempio `round(25.76543, 2)` restituisce `25.77` che è in l'approssimazione a due cifre decimali del numero `25.76543`). Se lo studente k non ha sostenuto esami, la lista in output in posizione k dovrà contenere il valore 0.
Ad esempio, se **M** è

$$\begin{pmatrix} 24 & 30 & 0 \\ 22 & 0 & 28 \\ 23 & 29 & 0 \\ 18 & 0 & 0 \end{pmatrix}$$

la funzione dovrà restituire un lista contenente (nell'ordine dato) i numeri `21.75, 29.5, 28.0`.

- **A_Ex3 (file)** In un file di testo sono presenti solo lettere (maiuscole o minuscole, no accenti), cifre (0,1,2,3,4,5,6,7,8,9), spazi bianchi e caratteri di fine riga ('\n'). Un numero è una parola di sole cifre, cioè è una sequenza costituita solo da caratteri dell'insieme {0,1,2,3,4,5,6,7,8,9} delimitata da spazi bianchi, caratteri di fine riga, l'inizio o la fine del file. Potete assumere che il file contenga solo informazioni corrette. Scrivete una funzione che legge il file e restituisce un dizionario le cui chiavi sono i numeri contenuti nel file, e per ogni chiave il valore è il numero di occorrenze della chiave (cioè del numero) nel file di testo. Ad esempio, se **file** contiene il testo:

```
La associazione A123 viene conosciuta anche come semplicemente
123 dal numero dei suoi 123 fondatori
Oltre i 123 fondatori ci sono altri 150 iscritti
Ciascuno di questi 150 iscritti paga una quota annuale di 1000
euro
Altri 500 euro sono dovuti per estendere i servizi della
associazione ai familiari
500 euro comprensivi di IVA
```

  La funzione deve restituire `{123:3,150:2,1000:1,500:2}`

- **A_Ex4 (file1,file2)** Scrivere una funzione che riceve in ingresso le informazioni sul ranking di alcuni giocatori di tennis ed il risultato dei loro incontri. Più precisamente, il file **file1** contiene il nome di un giocatore per riga, in ordine di ranking (il giocatore sulla prima riga è il numero 1 nel ranking, il giocatore sulla seconda è il numero 2, e così via). Si noti che per definizione non ci sono tennisti a pari merito nel ranking. Il file **file2** è in formato csv e contiene per ogni riga informazioni relative ad un incontro fra due giocatori, strutturate secondo il seguente schema:

```
Giocatore1, Giocatore2, set1, set2
```

  dove Giocatore1 e Giocatore2 indicano i nomi dei due giocatori dell'incontro, set1 e set2 indicano il numero di set vinti (un intero fra 0 e 3), rispettivamente dal Giocatore1 e dal Giocatore2. Ovviamente, chi ha vinto il maggior numero di set ha vinto l'incontro. Si assuma che ogni giocatore che compare in **file2** compaia anche in **file1**. La funzione deve restituire un dizionario avente come chiavi i nomi dei giocatori che hanno vinto almeno un incontro contro un giocatore che li precede nel ranking, e come valore di ciascuna chiave K un insieme contente tutti i nomi dei giocatori sconfitti da k (anche di quelli che li seguono nel ranking).
  Ad esempio, se il contenuto di **file1** è:

```
Djokovic
Nadal
Thiem
Federer
Medvedev
```

ed il contenuto di **file2** è

```
Federer, Medvedev, 3, 0
Thiem, Federer, 3, 2
Djokovic, Federer, 2, 3
Nadal, Federer, 3, 1
Nadal, Djokovic, 3, 2
```

La funzione deve restituire `{'Federer':{ 'Medvedev', 'Djokovic'}, 'Nadal':{ 'Federer', 'Djokovic'}}`