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

Compito d'esame -- 12 luglio 2019 -- Compito A

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 `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 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

- **A_Ex1(s1,s2)** Scrivere una funzione che prende in ingresso due stringhe, **s1** ed **s2**, e restituisce la loro distanza sintattica, calcolata secondo la seguente regola:
  (i) Se le due stringhe hanno la stessa lunghezza, la distanza sintattica coincide con il numero di caratteri diversi nella stessa posizione in **s1** ed **s2**. Ad esempio, se **s1**='pippo' ed **s2**='pappa', la funzione deve restituire 2, in quanto **s1** ed **s2** hanno la stessa lunghezza, il secondo ed il quinto carattere di **s1** ed **s2** sono diversi, e tutti gli altri caratteri sono uguali;
  (ii) Se invece **s1** è più lunga di **s2**, la distanza sintattica è ottenuta sommando la differenza fra le lunghezze di **s1** ed **s2** alla distanza sintattica fra **s2** e la stringa composta dai primi *n* caratteri di **s1,** dove *n* è pari alla lunghezza di **s2**. Ad esempio, se **s1**='acetone' ed **s2**='acero', allora la funzione deve restituire 3, in quanto la differenza fra le lunghezze delle due stringhe è 7-5=2, la stringa composta dai primi 5 caratteri di **s1** è 'aceto, e la distanza sintattica fra 'aceto' ed 'acero' è 1.
  (iii) Se **s2** è più lunga di **s1**, si procede in modo simmetrico rispetto al caso (ii).

- **A_Ex2 (M,s)** Scrivere una funzione che prende in ingresso una matrice **M** (rappresentata come lista di liste) contenente caratteri alfabetici, ed una stringa **s**. La funzione deve restituire `True` se **s** *è contenuta* in una riga o in una colonna di **M**. In caso contrario la funzione deve restituire `False`. Più precisamente, **s** è contenuta in una riga R di **M** se **s** è una sottostringa della stringa ottenuta concatenando tutti caratteri in R (dal primo all'ultimo). Analogamente per il contenimento in una colonna. Si assuma che **M** contenga almeno un elemento e che **s** sia non vuota. Ad esempio, se **M** è la matrice

```
a   m   o
r   e   a
i   d   i
a   x   a
```

ed **s**=’amo’ oppure **s**=’aria’ oppure **s**=’aia’, allora la funzione deve restituire `True` (infatti ‘amo’ è contenuta nella prima riga, ‘aria’ è contenuta nella prima colonna, ‘aia’ è contenuta nella terza colonna). Se invece s=’amore’ oppure s=’oca’, la funzione deve restituire `False`.
Suggerimento: si ricorda che per controllare se una stringa s1 è contenuta in una stringa s2 è possibile usare l’operatore booleano “in” scrivendo “s1 in s2”.

- **A_Ex3(file)** Scrivere una funzione che prende in ingresso un file di testo contenente numeri interi separati da spazi e/o andate a capo e restituisce un dizionario con chiave un intero *x* e valore la lista ORDINATA E SENZA RIPETIZIONI di tutti gli interi che *precedono immediatamente* ogni occorrenza di *x* nel file. Si noti che un intero *y* precede immediatamente un intero *x* se (leggendo il file in modo standard, cioè da sinistra a destra e dall’alto verso il basso), *y* viene prima di *x* ed i due numeri sono separati solo da spazi e/o comandi di andata a capo. Un intero *z* che compare solo all’inizio del file (e quindi non ha elementi che lo precedono) dovrà essere associato nel dizionario ad una lista vuota. Ad esempio, il file contiene:

```
1   3   24  3   12  8
8   7   3   12  24
2   24
```

  La funzione deve restituire il dizionario `{1 : [], 3: [1, 7, 24], 24:[2, 3, 12], 12:[3], 8:[8, 12] , 7:[8], 2:[24]}`.

- **A_Ex4(file**) Scrivere una funzione che prende in ingresso il nome di un **file** di testo contenente le performance di vendita di alcuni agenti di borsa nell’arco di una settimana. Più precisamente, il **file** contiene una riga per ogni giorno della settimana, e le righe sono ordinate dal lunedì al venerdì (cioè la prima riga corrisponde al lunedì, la seconda al martedì, ecc.). Ciascuna riga contiene una **sequenza di stringhe** (separate da “;”) nel seguente formato:

Agente-Importo

  dove **Importo** è la cifra totale incassata per le vendite effettuate in quel giorno dall’agente con nome **Agente**. Ogni **Agente** compare una sola volta in una riga. Se un **Agente** non ha effettuato vendite in un certo giorno, allora NON compare nella riga corrispondente.
  La vostra funzione deve costruire un dizionario con chiave il nome dell’agente e valore una lista ORDINATA contenente i giorni della settimana in cui l’agente ha avuto la migliore performance rispetto agli altri agenti (cioè i giorni in cui l’**Importo** a lui associato è risultato pari al massimo della giornata). Nel dizionario, i giorni della settimana devono essere indicati con gli interi da 1 a 5, dove 1 corrisponde al lunedì, 2 al martedì, e così via. Nella lista associata all’agente, *i giorni della settimana devono essere elencati in ordine crescente*. Si noti che in un certo giorno ci può essere più di un agente che ha ottenuto la performance migliore. Inoltre, *il dizionario deve contenere tutti gli agenti presenti nel file in input*, anche se questi non hanno mai avuto la performance migliore in alcun giorno. In questo caso, il valore del dizionario associato al nome sarà una lista vuota.

  Ad esempio, se il file contiene:

```
Mario-10000;Marco-3000;Anna-5000
     Mario-15000;Aldo-15000
          Anna-20000
Mario-5000;Marco-3000;Anna-6000
     Antonio-4000;Anna-3000
```

  allora la funzione deve restituire il dizionario `{'Mario': [1, 2], 'Marco': [], 'Anna': [3,4], 'Aldo': [2], 'Antonio': [5]}`