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

Compito d'esame -- 12 luglio 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(s1,s2)** Scrivere una funzione che prende in ingresso due stringhe, **s1** ed **s2**, e restituisce il loro grado di similitudine, calcolato secondo la seguente regola:
  (i) Se le due stringhe hanno la stessa lunghezza, il grado di similitudine coincide con il numero di caratteri uguali nella stessa posizione in **s1** ed **s2**. Ad esempio, se **s1**='lago' ed **s2**='mago', la funzione deve restituire 3, in quanto **s1** ed **s2** hanno la stessa lunghezza, il secondo, terzo e quarto carattere di **s1** ed **s2** sono uguali, mentre il primo carattere è diverso;
  (ii) Se **s1** è più lunga di **s2**, il grado di similitudine è ottenuto sottraendo 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** (si noti che il risultato può anche essere negativo). Ad esempio, se **s1**='acetone' ed **s2**='acero', allora la funzione deve restituire 2, 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', ed il grado di similitudine fra 'aceto' ed 'acero' è 4.
  (iii) Se **s2** è più lunga di **s1** si procede in modo simmetrico rispetto al caso (ii).

- **B_Ex2 (M,s)** Scrivere una funzione che prende in ingresso una matrice *quadrata* **M** (rappresentata come lista di liste) contenente caratteri alfabetici, ed una stringa **s**. La funzione deve restituire `True` se **s** *è contenuta* in una delle due diagonali di **M**. In caso contrario la funzione deve restituire `False`. Più precisamente, **s** è contenuta in una diagonale D di **M** se **s** è una sottostringa della stringa ottenuta concatenando tutti caratteri in D (dal primo all'ultimo, dove il primo è quello nella prima riga e l'ultimo quello nell'ultima riga). Si assuma che **M** contenga almeno un elemento e che **s** sia non vuota. Ad esempio, se **M** è la matrice (in rosso è evidenziata la diagonale principale, in verde quella secondaria)

```
a  m  o  r
r  i  o  y
i  s  a  x
```

ed **s**=’rosa’ oppure **s**=’aia’, allora la funzione deve restituire `True` (in quanto ‘rosa’ è contenuto nella diagonale secondaria e ‘aia’ in quella principale). Se invece s=’rosato’ oppure s=’oca’, la funzione deve restituire `False`.
Suggerimento1: per individuare gli elementi della diagonale secondaria potete sfruttare la seguente proprietà: in una matrice quadrata di dimensione *n* la somma degli indici di ciascun elemento della diagonale secondaria è sempre pari ad *n-1*, mentre per tutti gli altri elementi è sempre diversa da *n-1* (si sta assumendo di indicizzare righe e colonne a partire dallo 0, come in Python).
Suggerimento2: 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”.

- **B_ 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 *seguono immediatamente* ogni occorrenza di *x* nel file. Si noti che un intero *y* segue immediatamente un intero *x* se (leggendo il file in modo standard, cioè da sinistra a destra e dall’alto verso il basso), *y* viene dopo di *x* ed i due numeri sono separati solo da spazi e/o comandi di andata a capo. Un intero *z* che compare solo alla fine del file (e quindi non ha elementi che lo seguono) 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  5
```

La funzione deve restituire il dizionario `{1: [3], 3: [12, 24], 24: [2, 3, 5], 12: [8, 24], 8: [7, 8], 7: [3], 2: [24], 5: []}`.

- **B_Ex4(file**) Scrivere una funzione che prende in ingresso il nome di un **file** di testo contenente le performance di vendita di alcuni agenti immobiliari nell’arco di un mese di 31 giorni. Più precisamente, il **file** contiene una riga per ogni giorno del mese, ordinate da 1 a 31 (cioè la prima riga è relativa al giorno 1, la seconda al giorno 2, ecc.). Per i giorni in cui nessun agente ha venduto nulla, la riga è vuota, cioè include solo il `'\n'`. Ciascuna riga contiene una **sequenza di stringhe** (separate da “;”) nel seguente formato:

Agente-NumeroCaseVendute

dove **NumeroCaseVendute** è il numero di 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 contenente come primo elemento il massimo numero di case vendute (da quell’agente) in un giorno del mese (cioè la sua performance migliore), e come elementi successivi i giorni in cui l’agente ha avuto la sua migliore performance (cioè i giorni in cui il **NumeroCaseVendute** a lui associato è risultato per lui massimo in tutto il mese). Nel dizionario, i giorni del mese devono essere indicati con gli interi da 1 a 31. Nella lista associata all’agente, *i giorni in cui ha avuto la performance migliore (che possono essere più di uno) devono essere elencati in ordine crescente.*

Ad esempio, se il file contiene (per semplicità mostriamo solo le prime 10 righe del file):

```
Mario-1;Marco-3;Anna-5

Mario-1;Aldo-1

Anna-2


Mario-5;Marco-1;Anna-5

Antonio-4;Anna-3
```

allora la funzione deve restituire il dizionario `{'Mario': [5, 8], 'Marco': [3, 1], 'Anna': [5, 1, 8], 'Aldo': [1, 3], 'Antonio': [4, 10]}`. Infatti, la performance migliore di Mario si è avuto il giorno 8, in cui ha venduto 5 case, quella di Marco il giorno 1, in cui ha venduto 3 case, quella di Anna si è avuta nei giorni 1 e 8 (si noti che sono riportati in ordine crescente), in cui, in entrambi i giorni, ha venduto 5 case.