Compito d'esame -- 29 gennaio 2025 -- Compito A
Nota: gli studenti DSA devono sostenere gli esercizi 1, 2 e 4
Gli studenti che hanno superato la prova intermedia devono sostenere gli esercizi 2 e 3 e le ultime 4 domande dell'esercizio 4

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 il programma `REGISTRAstudente.py` 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.py`.

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.**

È 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.

Per risolvere l'esercizio si possono importare solo le seguenti librerie: numpy, math e re.

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,n)** Data una lista di interi **l** e un intero positivo **n**, scrivete una funzione che restituisca la posizione iniziale e finale della più lunga sottosequenza di **l,** lunga almeno 2, la cui differenza tra il valore massimo e il minimo è inferiore o uguale a **n**. Se ci sono più sottosequenze della stessa lunghezza, dovete restituire quella più a sinistra. Ad esempio, se **l** vale [**3, 5, 7**, 2, 8] ed **n** vale 4, la sottosequenza di lunghezza massima è quella dalla posizione 0 alla posizione 2. La funzione deve quindi restituire i 2 valori 0 e 2. Se la lista è vuota o la sottosequenza non esiste deve restituire -1 e -1.

- **Ex2(nomeFile)** Dato un file di testo contenente solo caratteri alfabetici minuscoli, spazi ' ', tab '\t' e newline '\n', calcolare un dizionario che ha come chiave un numero intero che rappresenta la lunghezza delle parole e come valore la stringa alfabeticamente più grande di quella lunghezza. Ad esempio, se il file contiene:

tanto va la gatta al lardo che ci lascia lo zampino
campa cavallo che l erba cresce
chi va a roma perde la poltrona

allora la funzione deve restituire il dizionario {5: 'tanto', 2: 'va', 3: 'chi', 6: 'lascia', 7: 'zampino', 1: 'l', 4: 'roma', 8: 'poltrona'}. Se il file non contiene nessuna parola la funzione deve restituire il dizionario vuoto.

- **Ex3(nomeFile, partenza, n)** Dato un file csv che contiene le informazioni sulle distanze e i tempi in minuti per andare da una città all'altra nel formato:

```
cittàPartenza, cittàArrivo, distanza, tempo
```

N.B. per ogni coppia di città esiste al massimo una riga di andata e al massimo una riga di ritorno.

Scrivere una funzione che prende in ingresso il nome di un file **nomeFile** nel formato di cui sopra, una città da cui partire **partenza** e un tempo **n** e calcola la città più distante dove si può andare e tornare nel tempo massimo (in minuti) indicato **n**. Se **partenza** non esiste o nessuna destinazione può essere raggiunta (andata e ritorno) nel tempo indicato allora la funzione deve restituire la stringa 'nessuna'. Notate che la distanza e il tempo per andata e ritorno POSSONO ESSERE DIVERSI, in questo caso, per distanza si intende il MASSIMO tra i 2 e per tempo di percorrenza la somma del tempo di andata e quello di ritorno. Se esistono più città alla stessa distanza massima raggiungibili da **partenza** la funzione deve restituire la città più piccola in ordine alfabetico. Per esempio, se il file contiene:

```
roma,firenze,320,120
roma,milano,660,180
firenze,roma,320,125
milano,roma,670,210
```

la città di partenza è 'roma' ed il tempo massimo è 400, la funzione deve restituire 'milano', poiché in 400 minuti si riesce ad andare e tornare da milano e firenze, ma milano è più lontana.

- **Ex4** Il file Ex4.py contiene la funzione veroFalso() che stampa 8 domande sullo schermo. La funzione deve essere modificata **cambiando solo il valore di ris nella riga 10**, elencando le lettere delle domande che ritenete essere vere. Ad esempio, se ritenete che le domande B e C sono vere la riga 10 deve essere modificata in

```
ris = 'BC'
```