Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica A.A. 2020-21

Compito d'esame – 17 Settembre 2021

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, aprite il programma Python `RegistraStudente.py` che si trova nella cartella `Esame` nell'ambiente IDLE. 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 la procedura.

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'esame (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.**

E' possibile consultare la documentazione ufficiale del linguaggio Python (file:///Library/Frameworks/Python.framework/Versions/3.9/Resources/English.lproj/Documentation/index.html), ma **non è possibile usare libri o appunti**. In caso di problemi tecnici chiedere ai docenti o ai tecnici del laboratorio.

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)** Scrivere una funzione che prende in input una lista **l** e restituisce la lista ottenuta da **l** eliminando l'elemento che viene ripetuto più volte in **l** (si assuma che sia sempre unico). Ad esempio, se **l**=[1,2,'pippo',1,2,1,'a','a'], allora la funzione deve restituire [2,'pippo',2,'a','a'], in quanto l'elemento che è ripetuto più volte è 1 (che è ripetuto tre volte, mentre 2 ed 'a' sono ripetuti due volte e 'pippo' compare una volta sola).

- **Ex2(M, I)** Scrivere una funzione che prende in input una matrice **M** rappresentata come lista di liste e contenente i valori 0 e 1, ed un insieme **I** contenente coppie (tuple di dimensione due) di interi non negativi che corrispondono a posizioni valide della matrice. La matrice **M** denota un dispiegamento di navi nel gioco "battaglia navale": la presenza del valore 1 in una posizione della matrice indica che quella posizione è occupata da (una porzione di) una nave, mentre lo 0 indica acqua. Le navi

sono dispiegate solo per righe (contrariamente al gioco tradizionale) e possono occupare una o più posizioni. In quest'ultimo caso, una nave è quindi rappresentata da una sequenza di 1 su una stessa riga. L'insieme **I** contiene le posizioni che sono colpite dall'avversario (ad esempio, se **I** contiene la coppia (0,2), allora la posizione identificata dalla riga 0 e dalla colonna 2 è colpita). La funzione deve restituire il numero di navi che sono affondate dall'avversario sulla base delle posizioni colpite. Si noti che una nave è affondata se tutti gli 1 che la denotano sono colpiti. Ad esempio, se **M** rappresenta la seguente matrice

011010
000110
000111

ed **I**={(0,1),(0,2),(1,3),(1,0),(2,3),(2,4),(2,5)}, la funzione deve restituire 2, in quanto le navi affondate dai colpi indicati in **I** sono due (rappresentate in rosso), mentre alcuni colpi sono andati in mare (1,0) o hanno colpito solo parte di una nave (1,3) (le posizioni colpite da questi colpi sono rappresentate in azzurro in figura). Si noti che in questo esempio la matrice contiene due navi sulla prima riga (una che occupa due posizioni ed una che occupa una sola posizione), una nave sulla seconda riga (che occupa due posizioni) ed una nave sulla terza riga (che occupa tre posizioni).

- **Ex3(p,a,file)** Vogliamo calcolare il modo più rapido per andare in aereo da una città ad un'altra facendo **al massimo uno stop**. Avete come input la città di partenza **p**, la città di arrivo **a** ed un file **file** con tutte le informazioni sui voli disponibili nel seguente formato:

CittàPartenza,CittàArrivo,TempoVolo

La funzione deve restituire il tempo minimo complessivo di volo da **p** ad **a** (cioè il tempo di volo, se il volo è diretto, altrimenti la somma dei tempi di volo dei 2 voli se c'è uno scalo). Se non c'è alcun modo per arrivare dalla città di partenza **p** a quella di arrivo **a** con al massimo uno stop, allora la funzione deve restituire la stringa 'Impossibile'. Ad esempio, se la città di partenza è Roma e la città d'arrivo è Parigi ed il file contiene:

Roma,Londra,150
Londra,Lisbona,150
Londra,Mosca,200
Roma,Mosca,180
Mosca,Lisbona,300
Roma,Londra,120
Mosca,Parigi,305
Londra,Parigi,60

allora il modo più rapido di andare da Roma a Parigi sarà con scalo a Londra con un tempo complessivo di volo di 180 (120+60). Notate che ci sono 2 voli da Roma a Londra ed ovviamente la soluzione minima usa il più breve dei 2. **Potete assumere che ogni volo abbia un tempo di volo sempre inferiore a 1000 minuti**.