

14 Giugno 2018

Esame di Fondamenti di Informatica per Ing. Gestionale

Compito B - Durata 1h45'

Istruzioni (leggere attentamente)

Note

La mancata osservanza delle seguenti regole può comportare la perdita di informazioni necessarie alla valutazione dell'esame.

Cartella di esame e registrazione dei dati dello studente

La cartella Esame contenente il compito da svolgere si trova sul Desktop. Entrare nella cartella e, prima di iniziare il compito, eseguire il programma `registrazione.pyc`. Inserire i dati personali fornendo (separatamente) *Numero di Matricola*, *Cognome* e *Nome*. Il programma genera un file `studente.txt` che non deve essere modificato manualmente. Verificare che i dati nel file `studente.txt` siano corretti; in caso di errore potete rieseguire il programma `registrazione.pyc`.

Svolgimento degli esercizi

Leggere attentamente il testo e risolvere gli esercizi proposti.

Per ogni esercizio avete una cartella `EsercN` che contiene un file dal nome `B_ExN.py` (dove N è il numero dell'esercizio) con lo scheletro della soluzione. Il file `B_S_ExN.pyc` contiene la soluzione precompilata dell'esercizio proposta dal docente. L'unico file da modificare è `B_ExN.py`. *Nel vostro interesse, non modificate altri file (ad esempio contenenti dati di test). Non create nuovi file.*

Per verificare la correttezza di un esercizio **DOVETE** usare il programma `TestEx.pyc`, che proverà la vostra soluzione su un certo numero di casi di test, mostrandovi il confronto tra i risultati ottenuti con la vostra soluzione e usando la soluzione proposta dal docente. Ciò ha lo scopo di facilitarvi nella ricerca e correzione di eventuali errori. *Ricordiamo comunque che in sede di correzione da parte dei docenti le vostre soluzioni verranno provate su dati di test diversi da quelli usati durante l'esame*

E' possibile consultare la documentazione ufficiale del linguaggio Python (ad esempio premendo F1 all'interno dell'ambiente di programmazione Idle), ma non è possibile usare libri o appunti. In caso di problemi tecnici chiedere ai docenti o ai tecnici del laboratorio.

Esercizi

1. **B_Ex1(s) (9 punti)** Completare la funzione Python `B_Ex1(s1, s2)` che prende in ingresso due stringhe `s1` e `s2`. La funzione deve restituire un numero intero che indichi la lunghezza del prefisso più lungo comune alle due stringhe. Ad esempio, se `s1 = "arciere"` e `s2 = "arco"`, la funzione dovrebbe restituire `3`. Se invece `s1 = "arciere"` e `s2 = "pippo"` la funzione dovrebbe restituire `0`. Se almeno una delle due stringhe è vuota la funzione deve restituire `0`.

2. **B_Ex2(m) (9 punti)** Completare la funzione Python `B_Ex2(m)` che riceve in ingresso una matrice `m` di interi rappresentata per righe come lista di liste e restituisce una lista contenente gli elementi della prima colonna di `m`. Se `m` è la matrice vuota la funzione deve restituire una lista vuota.

Esempio: se la matrice fosse

$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 5 & 2 \end{pmatrix}$$

la funzione dovrebbe restituire `[2, 3]`.

3. **B_Ex3(file) (9 punti)** Implementare la funzione `B_Ex3(file)` che prende in ingresso il nome di un file, contenente (uno per riga) i nomi delle squadre vincitrici di una competizione sportiva nei vari anni in cui si è disputata (albo d'oro). La funzione deve restituire la lista, *ordinata alfabeticamente*, delle squadre che hanno vinto più volte. Eventuali maiuscole devono essere ignorate, per cui ad esempio "roma" e "Roma" vanno considerate la stessa squadra. Si assuma invece che in tutti i casi in cui il nome di una squadra è composto da più parole (ad es. "Real Madrid"), queste siano sempre separate da uno spazio.

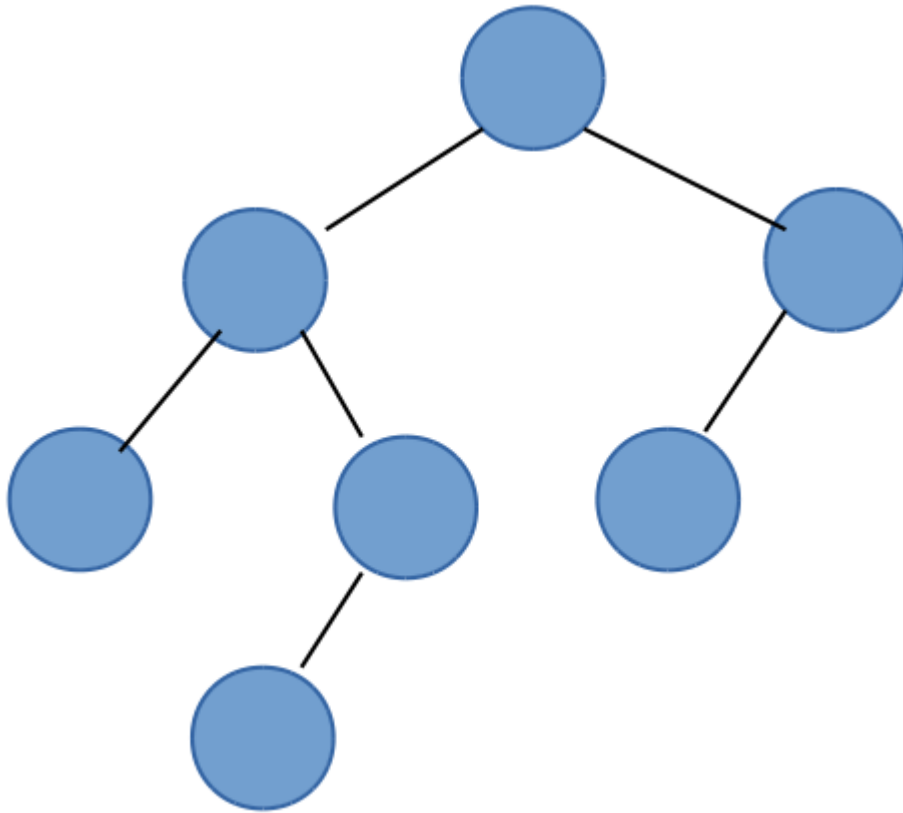
Esempio: se il file contenesse il seguente albo d'oro:

Barcellona\n real madrid\n real madrid\n valencia\n barcelona\n atletico madrid\n

allora la funzione dovrebbe restituire la lista `[barcellona, real madrid]`.

4. **B_Ex4(a) (5 punti)** Implementare la funzione `B_Ex4(a)` che riceve in ingresso un oggetto `a` della classe `BinaryTree`, che rappresenta un albero binario e la cui interfaccia è riportata sotto. La funzione deve restituire il numero di nodi **non** foglia (cioè i nodi con almeno un figlio) presenti nell'albero binario. Se l'albero è vuoto deve restituire il valore 0.

Ad esempio, se l'albero fosse quello della figura sottostante la funzione dovrebbe restituire il valore 4.



La classe `BinaryTree` implementa, tra gli altri, i metodi descritti dalla seguente interfaccia:

```
class BinaryTree:
    ## Il generico oggetto di questa classe rappresenta la radice di un (sotto)albero
    ## binario e contiene i riferimenti agli eventuali figli destro e sinistro
    def __init__(self, rootObj):
        ## Inizializza un albero. rootObj è l'identificatore della radice

    def getRightChild(self):
        ## Restituisce la radice del sottoalbero sinistro o None (se il nodo corrente
        ## non ha un figlio sinistro)

    def getLeftChild(self):
        ## Restituisce la radice del sottoalbero sinistro o None (se il nodo corrente
        ## non ha un figlio sinistro)

    def getRootVal(self):
        return self.key
        ## Restituisce l'identificatore associato al nodo rappresentato da self

    ## Altri metodi
```

Si noti che potreste non aver bisogno di tutti i metodi (o del costruttore) elencati sopra. Si noti anche che copia dei dati di test a partire dai quali sono costruiti gli oggetti passati alla funzione si trovano nei file `file1.txt ... file8.txt`. Non modificare per nessun motivo gli originali contenuti nella sottocartella `files`