

22 Ottobre 2018

Esame di Fondamenti di Informatica per Ing. Gestionale

Compito A - 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 `A_ExN.py` (dove N è il numero dell'esercizio) con lo scheletro della soluzione. Il file `A_S_ExN.pyc` contiene la soluzione precompilata dell'esercizio proposta dal docente. L'unico file da modificare è `A_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. **A_Ex1(l,c) (9 punti)** Implementare la funzione Python `A_Ex1(s1,s2,c)` che prende in ingresso una stringa `s1`, una stringa `s2` e un carattere `c`. La funzione deve restituire `True` se il numero di occorrenze di `c` in `s1` è uguale o maggiore del numero di occorrenze di `c` in `s2`. Se il numero di occorrenze di `c` in `s1` è minore del numero di occorrenze di `c` in `s2` oppure il carattere non appare in `s1` o `s2` la funzione deve restituire `False`.

Esempio: Se `s1 = "orsola"`, `s2 = "porro"` e `c = "o"` la funzione deve restituire `True` in quanto il carattere `o` appare 2 volte in `s1` e 2 volte in `s2`. Se invece `s1 = "orsola"`, `s2 = "porro"`, e `c = "r"` la funzione deve restituire `False`. La funzione deve restituire `False` anche se `s1` ed `s2` sono come sopra e `c="l"` oppure `c="z"` (nel primo caso `l` non compare in `s2`, nel secondo caso `z` non compare in nessuna delle stringhe in input alla funzione).

2. **A_Ex2(l) (9 punti)** Implementare la funzione Python `A_Ex2(l)` che prende in input una lista `l` contenente solo liste (eventualmente vuote) di stringhe e restituisce in output una lista contenente la stringa di lunghezza massima di ogni sottolista di `l`. *Le stringhe nella lista in output devono essere in ordine alfabetico*. Nel caso in cui in una sottolista ci sia più di una stringa di lunghezza massima, per questa sottolista la lista in output deve contenere solo la stringa di lunghezza massima che precede tutte le altre (cioè la prima incontrata scandendo la sottolista dal primo all'ultimo elemento). Nel caso in cui una sottolista della lista in input sia vuota, per questa sottolista non si deve inserire nella lista in output alcuna stringa. Infine, se la lista in input è vuota, la funzione deve restituire una lista vuota.

Esempio:

- Se la lista in input è `[["casa", "mamma"], ["alto", "abaco", "zorro"]]` la funzione deve restituire `["abaco", "mamma"]`
- Se la lista in input è `[["casa", "mamma", ""], [], ["alto", "abaco", "zorro"], []]` la funzione deve restituire `["abaco", "mamma"]`
- Se lista in input è `[]`, la funzione deve restituire `[]`

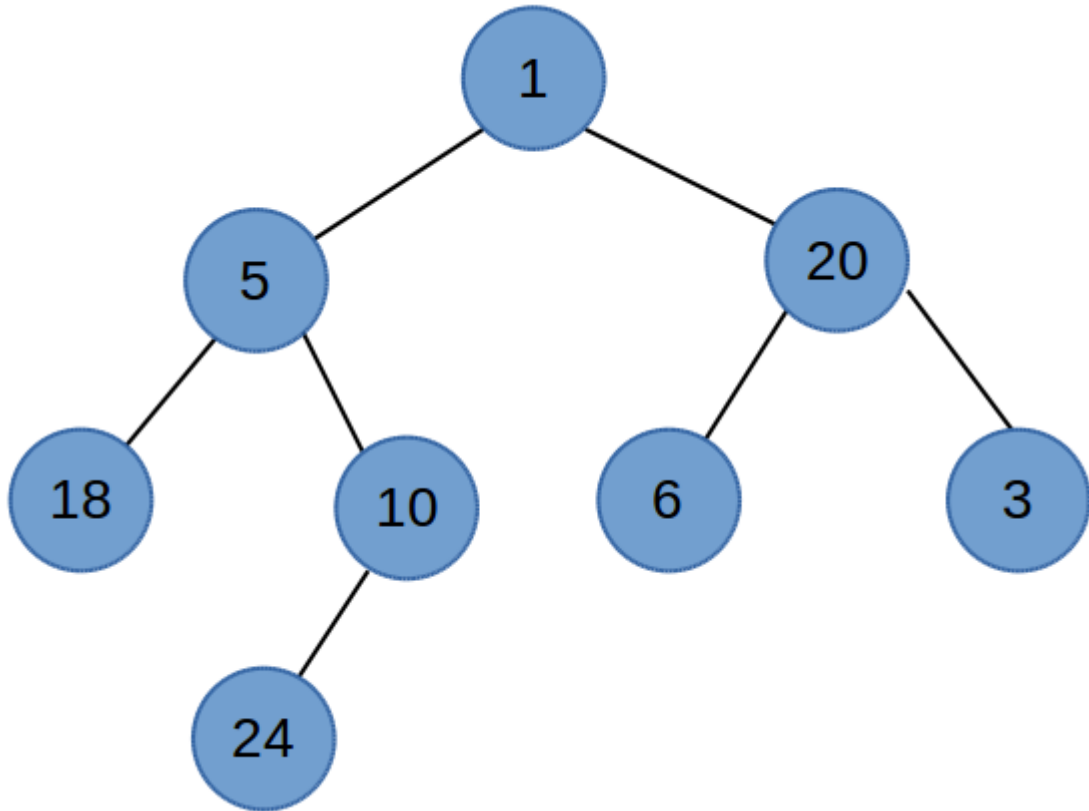
3. **A_Ex3(file) (9 punti)** Implementare la funzione Python `A_Ex3(file)` che prende in ingresso il nome di un file contenente la lista dei corsi superati da alcuni studenti ed il voto corrispondente nel seguente formato `'Codice_corso,Matricola_studente,Voto'`. La funzione deve restituire la matricola (come stringa) dello studente che ha ottenuto il massimo voto minimo, cioè che il voto minimo fra tutti gli esami che ha sostenuto è maggiore di tutti i voti minimi degli altri studenti. Se il file è vuoto la funzione deve restituire `None`.

Esempio: se il file contenesse il testo: `"C1,123,29\nC5,123,23\nC5,321,21\nC2,223,26"` allora la funzione dovrebbe restituire la stringa `'223'`.

Nota: Assumere nei dati contenuti nei file in input che lo studente con il massimo voto minimo sia sempre *uno solo*.

4. **A_Ex4(a, val) (5 punti)** Implementare la funzione Python `A_Ex4(a, val)` che riceve in ingresso un oggetto `a` della classe `BinaryTree` (la cui interfaccia è riportata sotto) e un valore intero `val`. Il primo parametro della funzione rappresenta un albero binario, i cui nodi hanno etichette di tipo stringa che rappresentano valori interi. La funzione deve restituire il numero di nodi il cui valore associato è *divisibile per* `val`. Se l'albero è vuoto la funzione deve restituire il valore 0.

Esempio: se l'albero fosse quello della figura sottostante e `val` avesse il valore 3 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 ... file5.txt`. Non modificare per nessun motivo gli originali contenuti nella sottocartella `files`.