Esercitazione 9

10 gennaio 2017

Termine per la consegna dei lavori: **martedì 17 Gennaio 2017** ore **23.59**.

Istruzioni

I lavori dovranno essere salvati in una cartella che deve contenere tutto e solo ciò che volete venga consegnato e valutato, dovrà essere **obbligatoriamente** un file **eseguibile** con estensione `.py` per **ognuno** degli esercizi. Controllate che l'esecuzione del comando:

```
python <nome_file>.py
```

per ognuno degli esercizi produca l'output desiderato.
Per consegnare gli elaborati dovete raggiungere la cartella contenente i file da inviare in modalità terminale (`cd path_della_cartella`) e quindi eseguire il comando:

```
consegna consegna9
```

verrà visualizzata la lista di tutto ciò che è stato inviato.
Consegne successive (entro il termine per la consegna) sovrascriveranno le precedenti, verrà valutata solo l'ultima consegna sottomessa.
È obbligatorio che all'interno di ogni file sia riportato il vostro nome, cognome e numero di matricola (riportati all'interno di una riga commento all'inizio del file, es: `#Mario Rossi 1234567`).

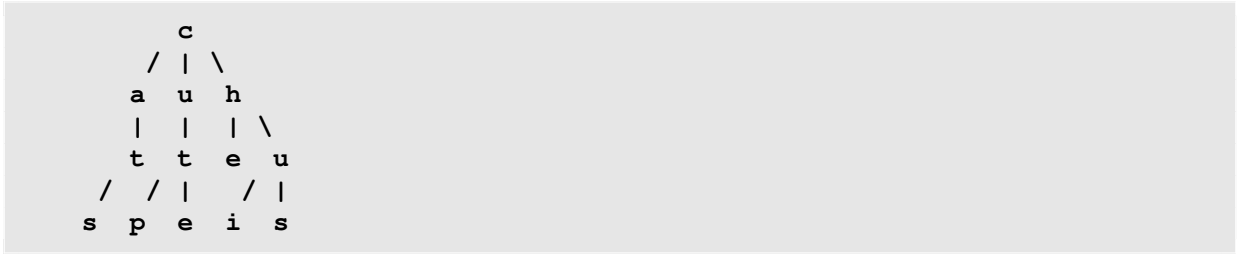
ATTENZIONE!

Gli unici moduli importabili ammessi in questa esercitazione sono i moduli `math` e `random`. Esercizi risolti utilizzando altri moduli importati riceveranno il punteggio minimo. Python contiene delle **built-in functions**, che potete utilizzare e la cui lista si può trovare a questo indirizzo: `https://docs.python.org/2/library/functions.html`.

Esercizio 1

Definire la classe TST (Ternary Search Tree) che rappresenta un albero ternario di ricerca. Questo tipo di albero viene spesso utilizzato come albero dei prefissi per fare ricerche incrementali su stringhe, quindi è pensato per contenere stringhe.

Un albero ternario di ricerca funziona come un albero binario ma con 3 figli anziché 2: ogni nodo dell'albero contiene un carattere (o è vuoto) e il figlio sinistro sarà associato a un carattere più piccolo (in termini alfabetici) e il figlio destro a un carattere più grande. Il figlio centrale invece conterrà la lettera successiva della parola.
Vediamo un esempio:



questo albero contiene al suo interno le seguenti parole: as, at, cup, cute, he, i, us.
Ovviamente con le stesse identiche parole si possono ottenere alberi ternari di ricerca di forma diversa perché dipende dall'ordine in cui sono state inserite.

Definire inoltre le seguenti funzioni **ricorsive**:

- una funzione che inserisce una parola all'interno dell'albero e lo ritorna con la parola inserita;
- una funzione che cerca se una parola esiste all'interno dell'albero;
- una funzione che ritorna il prefisso più lungo di una parola contenuto nell'albero. Nell' esempio il prefisso più lungo contenuto nell'albero della parola 'cutter' è 'cut';
- una funzione che conta il numero di nodi;
- una funzione che ritorna l'altezza dell'albero;
- una funzione (**non** ricorsiva) che carica un file di testo contenente solo parole scritte con caratteri alfabetici minuscoli e spazi, e ne crea l'albero ternario di ricerca associato.

Dare almeno un esempio di utilizzo delle funzioni create.
Per maggiori informazioni sugli alberi ternari di ricerca `https://en.wikipedia.org/wiki/Ternary_search_tree`.

Esercizio 2

Chiameremo albero binario natalizio, un albero binario *n-bilanciato* i cui nodi contengono le decorazioni. Un albero è n-bilanciato, quando, per ogni sotto-albero radicato in un suo nodo, il numero dei nodi del sotto-albero sinistro meno il numero dei nodi del sotto-albero destro è in valore assoluto al più 1.

Si chiede di creare una classe `XmasTree` (che è un albero binario) e le seguenti funzioni **ricorsive**:

- una funzione per stampare l'albero in modo formattato. Non viene richiesto nulla di artistico, si deve capire la struttura dell'albero e i valori nei nodi;
- una funzione `build_tree(decorations)` che accetta come parametro una lista di decorazioni (di qualsiasi tipo) e ritorna un albero binario natalizio con le decorazioni disposte nei suoi nodi (l'ordine di inserimento non è importante);
- una funzione `is_xmas_tree(tree)` che ritorna vero se l'albero passato come parametro è un albero binario natalizio (ovvero un albero n-bilanciato) altrimenti ritorna falso.

Se lo trovate necessario potete definire altre funzioni che possono aiutarvi nel risolvere i punti appena elencati. Dare almeno un esempio di utilizzo delle funzioni create.