Esercitazione 7

15 dicembre 2015

Apertura finestra per la consegna dei lavori: **mercoledì 16 dicembre** ore **12.00**
Termine per la consegna dei lavori: **martedì 29 dicembre** ore **23.59** (due settimane di tempo).

Istruzioni

I lavori dovranno essere salvati in una cartella che deve contenere tutto e solo ciò che volete venga consegnato e valutato, può essere 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 consegna7
```

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 `Matrice` che rappresenta una matrice di dimensione $n \times m$. In particolare la classe deve contenere:

1. un costruttore (`__init__`) con un parametro `values` che come valore di default vale `[[0]]` (matrice nulla 1×1) e in generale è una lista di liste rappresentanti le righe della matrice, ad esempio deve essere possibile istanziare una matrice con l'istruzione `M = Matrice([[1,2,3],[4,5,6]]);`

2. la definizione del metodo speciale `__repr__(self)` che ritorna una rappresentazione per la matrice, ad esempio quella sottostante;

   ```
   1 2 3
   4 5 6
   ```

3. la definizione dei metodi speciali:

   - `__add__(self, M)` che, data la matrice `M`, restituisce `None` se le matrici non hanno la stessa dimensione, altrimenti la matrice somma delle due;
   - `__sub__(self, M)` che, data la matrice `M`, restituisce `None` se le matrici non hanno la stessa dimensione, altrimenti la matrice differenza delle due;
   - `__mul__(self, M)` che, data la matrice `M`, restituisce `None` se la seconda matrice ha un numero di righe diverso dal numero di colonne della prima, altrimenti la matrice prodotto delle due;
   - `__pow__(self, e)` che, dato un naturale `e`, restituisce `None` se la matrice non sia quadrata, altrimenti la matrice potenza secondo l'esponente `e`;

4. la definizione delle funzioni (**non metodi**):

   - `salva(M, path)` che, data la matrice `M` e il percorso di un file, salva la matrice M all'interno del file (scegliete voi in quale formato);
   - `carica(path)` che, dato il percorso del file, legge la matrice al suo interno e la ritorna.

Per ognuno di questi punti dare almeno un esempio di invocazione.
Nota: i metodi `__add__`, `__sub__`, `__mul__` e `__pow__` vengono invocati, rispettivamente, con gli operatori `+`, `-`, `*` e `**`. Mentre l'operatore `__repr__` è invocato quando un oggetto di tipo `Matrice` viene stampato con `print`.

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 (come visto a lezione);
- 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.

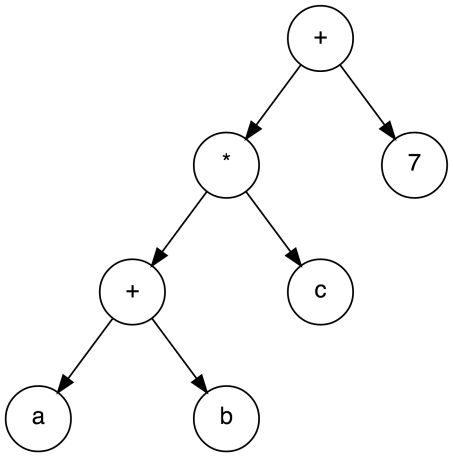
Se lo trovate necessario potete definire altre funzioni che possono aiutarvi nel risolvere i punti appena elencati. Dare almeno un esempio di utilizzo delle funzioni create.

Esercizio 3

Un'espressione matematica può essere rappresentata come un albero binario in cui i nodi rappresentano le operazioni e le foglie rappresentano gli operandi. Ad esempio l'espressione

$$(a + b)c + 7$$

può essere rappresentata dall'albero:



Scrivere una funzione `valuta(T)` che, dato un albero `T` che rappresenta una espressione, ne ritorni il suo valore. Si utilizzi la classe `Albero` costruita in laboratorio e tutte le funzioni di utilità che possono esservi utili.
Riportare almeno un esempio di invocazione su un albero che rappresenta un'espressione.