Esercitazione 5

24 novembre 2015

Termine per la consegna dei lavori: **giovedì 3 dicembre** 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, d'ora in poi sarà **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 consegna5
```

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

Scrivere una funzione `moneta` che simuli il lancio di una moneta, restituendo i valori "testa" oppure "croce", ciascuno con probabilità 0.5. Utilizzando la funzione precedente, scrivere la funzione iterativa `teste_consecutive` che simula il lancio della moneta finché non si ottengono `n` teste consecutive, stampando il numero di lanci che sono stati effettuati.

Similmente, scrivere la funzione ricorsiva `teste_consecutive_ric` (deve stampare gli stessi valori della funzione iterativa utilizzando però la ricorsione)

Esercizio 2

Data una sequenza `seq` (lista, stringa, ecc.) scrivere una funzione ricorsiva che generi la lista di tutte le permutazioni degli elementi di `seq`. Ad esempio, se `seq = 'uao'` la funzione deve ritornare la lista: `['uao', 'uoa', 'auo', 'aou', 'oua', 'oau']`. Ricordare che per generare le permutazioni di una sequenza è possibile fissare il primo elemento (al variare di esso nella sequenza) e generare tutte le permutazioni dei rimanenti.

Esercizio 3

Scrivere una funzione iterativa `frecce` che presa in input una stringa, la stampa nel modo seguente:
Fissata ad esempio la stringa uguale a "programmazione" (*caso pari*):
`p >> r >> o >> g >> r >> a >> m >> << m << a << z << i << o << n << e`
Oppure la stringa 'programma' (*caso dispari*):
`p >> r >> o >> g >> r << a << m << m << a`

Similmente, scrivere la funzione ricorsiva `frecce_ric` (deve stampare gli stessi valori della funzione iterativa utilizzando però la ricorsione)

Esercizio 4: Il gioco del 15

Link di spiegazione del gioco: `http://it.wikipedia.org/wiki/Gioco_del_quindici`
Tramite l'utilizzo di funzioni il programma deve:

1. Mostrare la tabellina di gioco ordinata e chiedere all'utente se vuole iniziare il gioco;
2. In caso di risposta affermativa, mescolare in modo casuale le tessere di gioco e mostrare la nuova tabella;
3. Chiedere all'utente quale mossa vuole effettuare;
4. Verificare se effettivamente la mossa sia corretta e nel caso affermativo modificare la tabella di gioco, altrimenti ripetere la richiesta;
5. Verificare se si è raggiunto lo stato finale (tab. ordinata), altrimenti ripetere tutti i passi dal punto 3.

Consiglio: per "disordinare" in modo corretto (quindi risolvibile) la tabella iniziale del gioco del 15, si possono effettuare delle mosse scelte in modo casuale ed effettuarle solo se sono mosse corrette.

Esercizio 5: Torre di Hanoi

Link di spiegazione del gioco: `https://it.wikipedia.org/wiki/Torre_di_Hanoi`
Considerare il caso di tre pile (A,B,C) e $n = 4$ dischi disposti inizialmente nella pila A.

Creare un gioco in cui l'utente deve riuscire a spostare i dischi dalla colonna A alla colonna C in un numero di mosse pari al minimo possibile $2^n - 1$.

1. Visualizzare le pile graficamente, per esempio nel modo seguente:
   ```
   Situazione iniziale:
   A: [4, 3, 2, 1]
   B: []
   C: []
   ```
   dove i numeri crescenti da 1 a 4 indicano i dischi dal più piccolo al più grande;

2. Chiedere all'utente quale mossa vuole effettuare, ossia che disco vuole spostare, da quale pila a quale altra;

3. Verificare se effettivamente la mossa sia corretta (si può spostare solo un disco alla volta e si può mettere un disco solo su un altro disco più grande, mai su uno più piccolo) e nel caso affermativo modificare le pile e stamparle, altrimenti ripetere la richiesta;

4. Verificare se si è raggiunto il numero di mosse massime $(2^n - 1)$, in tal caso stampare un `'Game Over :('` e chidere al giocatore se vuole vedere la soluzione o ricominciare il gioco;

5. In caso il giocatore voglia visualizzare la soluzione implementare l'algoritmo risolutivo mostrando graficamente tutti i passaggi della soluzione.

Consiglio: Creare le funzioni:

- `stampa_pile(...)` che stampa le pile;
- `sposta_un_disco(...)` che modifica le tre pile di dischi spostando un disco da una colonna ad un'altra;
- `sposta_piu_dischi(n,...)` che modifica le tre pile di dischi spostando n dischi da una colonna ad un'altra (dovrà essere richiamata ricorsivamente).