Recupero Verifica Ruby - 22 marzo 2016

Completa i seguenti esercizi.

Nell'elaborato mantieni i nomi delle classi e dei metodi del testo.

Verranno valutate sia la correttezza della soluzione sia il numero di righe di codice impiegate per scriverla.

Esercizio 1

```ruby
class Profile
  attr_accessor :name, :surname, :registration_date

  # requisiti perché un profilo sia considerato valido:
  # - registration_date deve essere un oggetto di classe Date
  # - name e surname devono essere stringhe:
  #   - non vuote
  #   - di lunghezza compresa tra 2 e 64
  #   - composto di soli caratteri minuscoli, ad eccezione del primo, maiuscolo
  def valid?
    false
  end

  # questo metodo restituisce un array di simboli.
  # Se l'oggetto è valido, restituisce un vettore vuoto
  # Se non lo è, per ogni attributo che non è valido, la chiave per
  # quell'attributo deve essere presente nel vettore, in qualsiasi ordine.
  # esempio: se surname e name non sono validi, restituisce [:surname, :name]
  def errors
    nil
  end
end
```

Puoi testare manualmente il comportamento della classe `Profile`

```ruby
require 'date'
u = Profile.new
u.name = 'User'
u.surname = 'Surname'
u.registration_date = Date.new(2011,2,4)
p u.valid?
p u.errors
```


© ICT4G FBK

Esercizio 2

Cerca di usare il minor numero di righe di codice possibile.

```
class Problem

  # dato in input un vettore di stringhe, restituisce la stringa che contiene
  # meno caratteri in maiuscolo. Se due o più stringhe hanno lo stesso numero
  # di caratteri in maiuscolo, restituisce una di queste.
  # Se il vettore e' vuoto, restituisce nil
  def self.min_uppercase(input)
    nil
  end
end
```

Esempi

```
p Problem.min_uppercase(['AA','AB','AC','AD','AE','Af']) # => 'Af'
p Problem.min_uppercase(['AA','AB','AC','AD','ae','af']) # => 'ae' oppure 'af'
p Problem.min_uppercase([]) # => nil
```

Esercizio 3

Cerca di usare il minor numero di righe di codice possibile.

```
class Problem

  # Dato in input un oggetto, restituisce la profondità dell'oggetto, che è
  # cosi definita:
  # (caso base) se un oggetto non è di classe Array, ha profondità 0;
  # (ricorsione) se un oggetto è di classe Array, ha profondità 1 + x,
  # dove x è la profondita' massima tra tutti gli elementi del vettore. Se il
  # vettore non ha elementi, allora x = 0
  def self.max_depth(input)
    0
  end
end
```

Esempi

```
p Problem.max_depth(nil) # => 0
p Problem.max_depth(123) # => 0
p Problem.max_depth([1,2,3]) # => 1
p Problem.max_depth([]) # => 1
p Problem.max_depth([[1],2,3]) # => 2
p Problem.max_depth([[],1,2,3]) # => 2
```


© ICT4G FBK