Introducción a la programación

Guía 8: Pilas y colas

1c2025

Guía 7 - Ejercicio 6.4

`problema columnas_ordenadas` (in m:$seq\langle seq\langle \mathbb{Z} \rangle \rangle$) : $seq\langle$Bool$\rangle$
{
  `requiere:` { $esMatriz(m)$ }
  `asegura:` { Para todo índice de columna c: $0 \leq c < |m[0]| \rightarrow$
    $(res[c] = true \leftrightarrow ordenados(columna(m, c)))$ }
  `asegura:` {$|res| = |m[0]|$}
}

Nota: Reutilizar la función `ordenados()` (guía 7, ejercicio 1.6) implementada previamente para listas

Guía 7 - Ejercicio 6.4 - Casos de Test

Elaboremos casos de test para la solución. ¿Qué casos de test podemos pensar?

Caso	**Entradas**	**Esperado**
Es la matriz más pequeña	m tiene sólo el valor 0	Verdadero
Es un vector no ordenado	única columna: 5 -1 0	Falso
Es una única fila	única fila: -5, 0, -10, 20	Verdadero, Verdadero, Verdadero, Verdadero

Además podemos pensar más casos: la columna ordenada es la primera, la última o está en el medio, o bien están todas ordenadas menos una columna que esta primera, última o en el medio.

Guía 8 - Ejercicio 1.1

Implementar una función
generar_nros_al_azar(in cantidad : int, in desde : int,
in hasta : int) → Pila[int].
Pueden usar la función random.randint(< desde >, < hasta >)
y la clase LifoQueue() que es un ejemplo de una implementación básica:

```
from queue import LifoQueue as Pila

p = Pila()
p.put(1) # apilar
elemento = p.get() # desapilar
p.empty() # vacia?
```

Guía 8 - Ejercicio 1.1

`problema generar_nros_al_azar` ($in\ cantidad : \mathbb{Z}$, $in\ desde : \mathbb{Z}$, $in\ hasta : \mathbb{Z}$) : Pila[$\mathbb{Z}$] {

`requiere:` $\{cantidad \geq 0\}$

`requiere:` $\{desde \leq hasta\}$

`asegura:` {El tamaño de $resultado$ es igual a $cantidad$}

`asegura:` {Ningún elemento de $resultado$ es menor a $desde$ ni mayor a $hasta$}

}

Guía 8 - Ejercicio 1.1

Elaboremos casos de test para la solución. ¿Qué debe cumplir la pila generada? ¿Qué casos de test podemos pensar?

Caso	**Entradas**	**Esperado**
La cantidad es 0	cantidad=0 desde=0 hasta=10	Pila vacía
La cantidad es 10	cantidad=10 desde=0 hasta=10	Pila con 10 elementos
El rango es desde 10 hasta 20	cantidad=10 desde=10 hasta=20	Pila con elementos entre 10 y 20

Guía 8 - Ejercicio 1.3

`problema buscar_el_maximo` $(in\ p : Pila[\mathbb{Z}]) : \mathbb{Z}\ \{$

`requiere:` $\{p$ no está vacía$\}$

`asegura:` $\{resultado$ es un elemento de $p\}$

`asegura:` $\{resultado$ es mayor o igual a todos los elementos de $p\}$

$\}$

Guía 8 - Ejercicio 1.3

Elaboremos casos de test para la solución.

Caso	**Entradas**	**Esperado**
Pila con un solo elemento	Pila([7])	7
Pila con varios elementos	Pila([1,2,3,4,5])	5
Pila con números negativos	Pila([-1,-2,-3])	-1

¿Qué otros casos de test podríamos probar?

Guía 8 - Ejercicio 2.13

Bingo: un cartón de bingo contiene 12 números al azar en el rango $[0, 99]$.

1. implementar una función $armar\_secuencia\_de\_bingo$
2. implementar una función $jugar\_carton\_de\_bingo$

```
from queue import Queue as Cola
#importa Queue y le asigna el alias Cola

c = Cola() #creo una cola
c.put(1) # encolo el 1
elemento = c.get() # desencolo
c.empty() # devuelve true si y solo si
          # la cola está vacía
```

2.13.1

problema armar_secuencia_de_bingo () : $Cola[\mathbb{Z}]$ {
  requiere: {True}
  asegura: {$resultado$ solo contiene 100 números del 0 al 99 inclusive, sin repetidos}
  asegura: {Los números de $resultado$ están ordenados al azar}
}

2.13.2

`problema jugar_carton_de_bingo`
($in\ carton : seq\langle\mathbb{Z}\rangle, in\ bolillero : Cola[\mathbb{Z}]$) : $\mathbb{Z}$ {
  `requiere:` {$carton$ solo contiene 12 números, sin repetidos, con valores entre 0 y 99 inclusive}
  `requiere:` {$bolillero$ solo contiene 100 números del 0 al 99 inclusive, sin repetidos}
  `asegura:` {$resultado$ es la cantidad mínima de jugadas necesarias para que todos los números del $carton$ hayan salido del bolillero}
}

Guía 8 - Ejercicio 2.13.2

Elaboremos casos de test para la solución.

Caso	**Entradas**	**Esperado**
El jugador gana en las primeras 12 jugadas	carton=[0,...,11] bolillero=Cola([0,...,11])	12
El jugador gana en la última jugada	carton=[0,...,11] bolillero=Cola([49,...,0])	50

¿Qué otros casos de test podríamos probar?