Programación – Certamen 1 - Jueves 3 de Octubre de 2024

Contexto

Los colibríes (picaflores) de Pythonia son aves muy organizadas. Antes de salir a comer cada día hacen un plan riguroso de cómo van a recorrer la plantación de flores donde se alimentarán, y tienen muy claro cuál es la ingesta mínima de flores que deben alcanzar.

Una plantación de flores es un área unidimensionales dividida en casillas. Su configuración se almacena en un *string* que registra la cantidad de flores en cada casilla. Por ejemplo, la siguiente plantación tiene 5 flores en la primera casilla, ninguna en la segunda, 4 en la tercera, y así sucesivamente:

```
"5042132"
```

Por su parte, el plan de alimentación también se encuentra almacenado en un *string*, en el que se espefican los movimientos que hace el ave, incluyendo la dirección (izquierda o derecha) y la cantidad de flores que planea consumir al llegar a la casilla de destino. Por ejemplo, en el siguiente plan el picaflor se desplaza desde una posición inicial 3 casillas a la derecha y al llegar desea comer 5 flores, posteriormente se desplaza 5 casillas a la izquierda para consumir 2 flores, y así sucesivamente:

```
"3D5;5I2;4D1;2D9;9I3"
```

Todos los números utilizados, tanto los que indican desplazamiento como la cantidad de flores a comer, **son de un dígito**.

Preguntas

1. **[20 %]** En la hoja de respuesta correspondiente, realice el ruteo del programa que se incluye a continuación. Además, indique en el recuadro de la pantalla lo que imprime como salida.

Cada vez que el valor de una variable cambie, escríbalo en una nueva fila de la tabla.

Si una variable es de tipo *string*, su valor debe ir entre comillas. La tabla tiene suficientes filas.

Programa para el ruteo

```
def promediar_comidas(plan):
    plan = plan + ";"
    s = 0
    n = 0
    i = 0
    while i<len(plan):
        f = plan[i:i+3]
        if f[1] == "I" or f[1] == "D":
            if f[2] in "0123456789":
                p = int(f[2])
                s += p
            else:
                return -1
        else:
            return -1
        i += 4
        n += 1
    return round(s/n)

prom = promediar_comidas("3D5;5I2;4D1")
print("Promedio:", prom)
prom = promediar_comidas("D5;3D2")
print("Promedio:", prom)
```

Programación – Certamen 1 - Jueves 3 de Octubre de 2024

Nombre

Rol — Paralelo

Programa	Función promediar_comidas					
prom	plan	s	n	i	f	p

Salida (Pantalla):

Programación – Certamen 1 - Jueves 3 de Octubre de 2024

Nombre

Rol — Paralelo

2.

Programación – Certamen 1 - Jueves 3 de Octubre de 2024

Nombre														
Rol										—		Paralelo			

3.

Programación – Certamen 1 - Jueves 3 de Octubre de 2024

2. **[40 %]** Escriba la función **`comer(plantación, plan, inicio)`**, que recibe como parámetro un *string* que contiene la configuración de una plantación, un *string* que contiene el plan del picaflor y un número entero que indica la casilla de la plantación donde comenzará el recorrido. La función debe ejecutar, en la medida de lo posible, el plan de alimentación, manteniendo actualizado el suministro de flores conforme se vayan comiendo. Si se quiere comer de una casilla en la que no hay suficientes flores, se consumirán sólo las existentes. Si se quiere desplazar fuera de los límites de la plantación, debe ajustarse el movimiento a la última casilla válida. Por ejemplo, si se quiere ir más a la izquierda de la casilla 0, debe detenerse en la 0. Una vez concluído el plan, la función debe retornar la cantidad efectiva de flores que pudo comer el ave.

Puede suponer que los *strings* recibidos como parámetro cumplen con el formato indicado en el contexto.

Nota: Si bien la función `promediar_comidas(plan)` del ruteo no es útil directamente, comprenderla puede ayudarle a resolver esta pregunta.

Ejemplos:

```
>>> print(comer("5042132", "3D5;5I2;4D1;2D9;9I3", 2))
11
>>> print(comer("111111", "1D2;1D2;1D2;1D2;1D2;5I2", 0))
6
```

Estudie cuidadosamente los ejemplos antes de comenzar a desarrollar su solución.

3. **[40 %]** Escriba un programa que solicite inicialmente el plan de comidas y la ingesta mínima de alimento que debe satisfacer el picaflor. Posteriormente, debe solicitar la cantidad de plantaciones que serán analizadas. Para cada una de las plantaciones se debe pedir la configuración de las flores e indicar si cumple o no con la ingesta mínima.

Al finalizar, se debe mostrar cuál es la plantación que permitiría obtener la mayor cantidad de alimento al picaflor. En caso de que haya empate en el mayor, puede mostrar cualquiera de los que empatan. Guíese por el ejemplo a continuación.

Ejemplo:

```
Ingrese el plan de comida: 3D5;5I2;4D1;2D9;9I3
Ingrese la ingesta mínima para el día: 11
Ingrese el número de plantaciones a analizar: 3

Plantación 1
Ingrese la plantación: 5042132
Ingrese la casilla de inicio: 2
Cumple! Come 11

Plantación 2
Ingrese la plantación: 1111111111
Ingrese la casilla de inicio: 0
No cumple con la ingesta mínima, apenas come 4

Plantación 3
Ingrese la plantación: 6271438001
Ingrese la casilla de inicio: 1
Cumple! Come 17

La plantación 3 es la que permite comer más.
```

Nota: Puede suponer que la función `comer(plantación, plan, inicio)` de la P2 se encuentra implementada y funciona correctamente. No es necesario volver a copiarla y puede utilizarla aunque no haya respondido correctamente la pregunta 2. Además, puede suponer que todos los *strings* ingresados cumplirán con el formato descrito en el contexto.