Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

CI2691 - Laboratorio de Algoritmos y Estructuras I

Enero-Marzo 2024

Laboratorio (Semana 3)

(5 puntos)

1. Descripción de la Actividad

Se quiere que realice programas en Python que resuelvan los siguientes problemas. Para cada programa se le proporcionará la firma de una función, la cual deberá completar. Todas las soluciones deberán estar en un solo script `LabSem3.py`. No serán necesarias verificaciones del input, es decir, si el requerimiento indica que se recibirá un número entero positivo, entonces solo se probarán números enteros positivos.

- Dado un número entero positivo, se quiere saber si es o no un número primo. Un número es primo si solo tiene dos divisores: él mismo y uno.

  Firma de la función:

  ```python
  def es_primo(num: int) -> bool:
      ...
  ```

  Ejemplos de ejecución:

  ```python
  es_primo(10) # False
  es_primo(2)  # True
  ```

- Dado un número entero positivo `num`, se debe computar la suma del cuadrado todos los números impares en el intervalo `[0..num)`.

  Firma de la función:

  ```python
  def suma_cuadrado_impares(num: int) -> int:
      ...
  ```

  Ejemplos de ejecución:

  ```python
  suma_cuadrado_impares(5) # 10
  suma_cuadrado_impares(6) # 35
  ```

- Dado dos números enteros positivos `m` y `n`, se debe computar si ambos números son amigos. Dos números son amigos si la suma de los divisores propios de cada uno es igual al otro número. Por ejemplo, los números 220 y 284 son números amigos. Los divisores propios de 220 son 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 y 110, cuya suma es 284. Por otro lado, los divisores propios de 284 son 1, 2, 4, 71 y 142, cuya suma es 220.

Firma de la función:

```
def son_amigos(m: int, n: int) -> bool:
    ...
```

Ejemplos de ejecución:

```
son_amigos(220, 284) # True
son_amigos(100, 8)   # False
```

- Dada una lista de números reales, se quiere saber si la lista está ordenada en orden no decreciente.

  Firma de la función:

  ```
  def esta_ordenado(numeros: List[float]) -> bool:
      ...
  ```

  Ejemplos de ejecución:

  ```
  esta_ordenado([0.0, 1.5, 3.0, 10.0])  # True
  esta_ordenado([0.0, 1.5, 3.0, -1.0])  # False
  ```

- Dada una lista de números reales y un número real `n`, se debe retornar la posición en la lista donde se encuentre la primera instancia de `n`. En caso de que la lista no tenga el número `n` se debe retornar -1.

  Firma de la función:

  ```
  def buscar(numeros: List[float], n: float) -> int:
      ...
  ```

  Ejemplos de ejecución:

  ```
  buscar([-5.0, 100.5, 42.0, 73.37], 42.0)  # 2
  buscar([-5.0, 100.5, 42.0, 73.37], 7.0)   # -1
  ```

- Dada una lista no vacía de números reales, se quiere calcular el promedio de los mismos.

  Firma de la función:

  ```
  def promedio(numeros: List[float]) -> float:
      ...
  ```

  Ejemplos de ejecución:

  ```
  promedio([0.0, 1.0, 2.0])  # 1.0
  promedio([5.0, 10.0])  #  7.5
  ```

- Dado un string que representa a una palabra, se quiere saber si la misma es palíndromo o no. *Hint: Se puede acceder al i-ésimo caracter de un string en Python usando la sintaxis de acceso a lista.* Por ejemplo:

  ```
  palabra: str = "Hola!"
  print(palabra[0]) # H
  ```

Firma de la función:

```
def es_palindromo(palabra: str) -> bool:
    ...
```

Ejemplos de ejecución:

```
es_palindromo("talenelat") # True
es_palindromo("shallan") # False
```

- Dada una matriz $M \times N$ de números enteros, se quiere encontrar el valor de la suma del máximo de cada fila.

  Firma de la función:

  ```
  def suma_maximo(matriz: List[List[int]]) -> int:
      ...
  ```

  Ejemplo de ejecución:

  ```
  suma_maximo([
      [0,  1,   2,  3,  4],
      [10, -10, 5,  8,  3],
      [-5, -6, -7, -8, -9]
  ]) # 9
  ```

- Dada una matriz $N \times N$ de números enteros, se quiere determinar si la matriz es simétrica o no. Una matriz M es simétrica si para cada par de índices i, j tal que $0 \leq i \leq j < N$ se cumple que $M[i][j] = M[j][i]$.

  Firma de la función:

  ```
  def es_simetrica(matriz: List[List[int]]) -> bool:
      ...
  ```

  Ejemplo de ejecución:

  ```
  es_simetrica([
      [0, 1, 2],
      [1, 1, 5],
      [2, 5, 2]
  ]) # True
  es_simetrica([
      [0, 1, -1],
      [1, 1, 5],
      [2, 5, 2]
  ]) # False
  ```

2. Evaluación

Se implementará un script para la evaluación de cada solución propuesta, el cual aplicará una variedad de casos de prueba, y generará un resultado final. En conjunto con este enunciado, se proporcionará el script `light_tests.py`. Este actúa como una versión reducida del evaluador auténtico, es decir, contiene un número menor de casos de prueba y estos son de menor complejidad. Este instrumento les facilitará la comprensión de cómo se llevará a cabo la evaluación real y les permitirá

adaptar sus soluciones de manera adecuada. Aunque no se realizará una evaluación del estilo de codificación, se efectuarán correcciones en este aspecto, lo cual les será de utilidad en evaluaciones futuras, donde dicho estilo podría ser objeto de evaluación.

Para correr `light_tests.py` necesitas colocarlo junto al script del laboratorio en un mismo directorio. Luego se ejecuta:

```
python light_tests.py
```

3. Condiciones de Entrega

El programa correspondiente al laboratorio `LabSem3.py` y la declaración de autenticidad debidamente firmada, deben ser incluidos en un archivo comprimido, en formato `tar.xz` (o `zip`), denominado `LabSem3_X.tar.xz` (o `LabSem3_X.zip`), donde `X` representa el número de carné del estudiante. La entrega del archivo debe realizarse a través de Gmail, enviándolo al correo electrónico **16-10072@usb.ve** antes de las 11:59 PM del día domingo 04 de febrero de 2024. Las entregas que se realicen después de la hora límite tendrán una penalización de un (1) punto, más un punto adicional por cada hora de retraso. Por lo tanto, después de 4 horas de retraso, la evaluación será anulada.