FUNDAMENTOS DE COMPUTACIÓN

Grado en ingeniería en tecnologías industriales

Universidad de Cantabria

Examen Extraordinario (Aula)

7 de septiembre del 2019

Nombre y apellidos:

---

1. Supongamos que `L` es una lista de números naturales con longitud un número par n. Se dice que `M` es la lista *intercambio-par-impar* de `L` si tiene la misma longitud n, y para todos los indices pares i con $0 \leq i < n-1$ se tiene que

   ```
   M[i]  == L[i+1]  and M[i+1] == L[i]
   ```

   es `True`. Por ejemplo, si

   ```
   L = [30, 50, 70, 90, 60, 40]  entonces  M = [50, 30, 90, 70, 40, 60]
   ```

   es la lista *intercambio-par-impar* de `L`. Completar la siguiente función para que devuelva como se especifica:

   ```
   def  IntercambioParImpar(L):
        """Devuelve una lista que es el intercambio-par-impar de L. No modificar L.
               PreC: L es una lista con valores enteros y, su longitud es par y mayor que cero. """
   ```

2. Se ejecutan los siguientes programas :

   f.py

   ```
   1  def f(x,y):
   2        y += 1
   3        x[y] += 1
   4
   5  A = {1:0, 2:0}
   6  B = 1
   7  f(A,B)
   ```

   g.py

   ```
   1  def g(a,b):
   2        if b:
   3            b.append('w')
   4        a += b
   5        return b
   6  w = 3; x = 0
   7  y = [4,3]; z = [5,4]
   ```

   ¿Cuál es la salida de `f.py` y `g.py` ?

`print(A)`
 	
`print(B)`
	`print(g(w,x))`
	
	`print(g(y,z))`
	
	`print(w)`
	
	`print(y)`

3. Supongamos que las variables B1 y B2 tienen un valor booleano. Escribir una expresión de valor booleano que sea `True` si y solo si exactamente una de las variables B1 y B2 es `True` (es decir, una es `True` y la otra `False`). La expresión debería ser `False` en caso contrario.

   `SOLUCIÓN:`

4. ¿Cuál es el resultado de la ejecución del siguiente programa? Si se produce un error describirlo.

```
x = [10, 20]; y = [30, 40]
temp = x
x = y                       |
y = temp                    |
print(x[0], x[1])           | SOLUCIÓN
print(y[0], y[1])           |
print(temp[0], temp[1])     |
z = x[0]
print(z)
```

5. Supongamos que disponemos de la clase `class Punto` con atributos números x e y para las coordenadas, y los siguientes métodos:

```
def __init__(self,x,y):
  """ Crear un Punto."""
def Dist(self, otro):
  """ Devuelve la distancia  del punto self al punto  otro."""
def VecinoAleatorio(self):
   """ Devuelve un Punto aleatorio con distancia <= 2 al Punto self """
```

   El siguiente programa simula un robot que comienza en $(0, 0)$ y salta aleatoriamente de un punto a otro en el plano.

```
P = Punto(0,0)
Z = P
t = 0
while P.Dist(Z) < = 100 and t < 100000:
    P = P.VecinoAleatorio()
    t += 1
```

   1. ¿Es posible que justo después de que termine este código t sea inferior a 100000? Explica tu respuesta en no más de tres lineas:

   2. Supongamos que cuando el robot realiza un salto desde el Punto P a un Punto con distancia mayor que 1, *tu hermano pequeño* tira del robot para que regrese al punto P. Escribir un código que simule este proceso para el robot comenzando en (0,0) e intentando 100 saltos.

6. ¿ Qué se muestra por pantalla después de ejecutar este programa ?

```
1  L = [1,2,3,4,5,6,7,8,9]; i = 0; s = ''
2  while i < len(L):
3    if i % 2:
4        i += 1
5        continue
6    elif i > 8:
7        break
8    s += str(L[i])
9    i += 2
10 else:
11   print("ejecutado")
12 print(s)
```

SOLUCIÓN

7. A continuación aparecen tres funciones f, g y h que toman como argumento una lista L de números de longitud n.

```
def f(L)
    s = 0
    for k in range(len(L)):
        j = 1
        while j < k:
            s += L[j]
            j *= 3
    return s
```

```
def g(L):
    for  valor in L:
          if (valor %2 == 1):
               valor += 1
    return valor
```

```
def h(L):
    n = len(L); i = 0
    while n >= 1:
         n -= 2**i
         i = i+1
    return i
```

¿Cuál de los siguientes afirmaciones describe con mayor precisión cómo crece el tiempo de ejecución de cada una de ellas a medida que n crece? *No contestar al azar, se penaliza con 0,2 puntos cada respuesta incorrecta*

(a) Crece linealmente, como n
(b) Crece cuadráticamente, como n^2
(c) Crece menos que linealmente
(d) Crece más que cuadráticamente

SOLUCIÓN para f:	SOLUCIÓN para g:	SOLUCIÓN para h:

8. Queremos contar cuántas veces aparece cada carácter en una cadena, para ello se pide construir una función `histograma` que tenga como argumento una cadena y devuelva el diccionario con caracteres como `claves` y contadores como los `valores` correspondientes. La primera vez que vea un carácter, agregará un elemento al diccionario. Después de eso, incrementaría el valor de un elemento existente.

```
>>> histograma('abracadabra')
{'a': 5, 'b': 2, 'r': 2, 'c': 1, 'd': 1}
```

```
def histograma(cad):
```

9. Se consideran los números en binario $N = 1100110000101$ y $M = 111011010011$. Calcular:

   - La suma, resta de $N + M$ y $N - M$.

     `N+M`

     `N-M`

   - División euclídea(cociente y resto) de N entre M.

     `Cociente:`

     `Resto:`

   - Convertir a octal N y a hexadecimal M.

     `Octal N:`

     `Hexadecimal M:`

10. En informática la distancia de **Hamming** mide la efectividad de los códigos dependiendo de la diferencia entre una palabra de código válida y otra. A esta diferencia se le llama distancia de Hamming, y se define como el número de bits que tienen que cambiarse para transformar una palabra de código válida en otra palabra de código válida. Si dos palabras de código difieren en una distancia d, se necesitan d errores para convertir una en la otra.

    Por ejemplo:

    - La distancia Hamming entre `'1011101'` y `'1001001'` es 2.
    - La distancia Hamming entre `'2143896'` y `'2233796' es 3.`
    - La distancia Hamming entre `'tener'` y `'reses'` es 3.

    Se denomina así gracias a su inventor Richard Hamming(1998), profesor de la Universidad de Nebraska.

    Se pide implementar dos funciones, a saber, una iterativa `hamming_iter` y otra recursiva `hamming_recur` teniendo como argumento dos cadenas `cad1` y `cad2` de la misma longitud y devolviendo la distancia de Hamming entre las dos cadenas.

    ```
    def hamming_iter(cad1, cad2):
    ```

    ```
    def hamming_recur(cad1, cad2):
    ```

Si saben de mi trabajo, no necesitan mi currículum. Si necesitan mi currículum, no conocen mi trabajo. Grigori Perelman(2010) en una entrevista al rechazar un millón de dólares por resolver uno de los problemas del milenio.