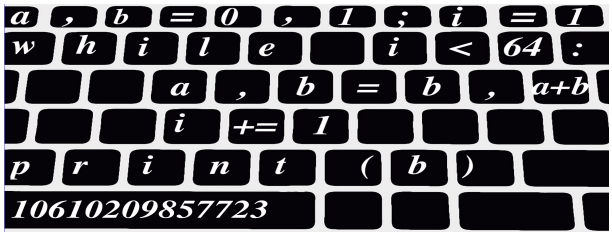

FUNDAMENTOS DE COMPUTACIÓN
Grado en ingeniería en tecnologías Industriales
Universidad de Cantabria

Examen Evaluación Continua (parte escrita)
26 de noviembre del 2021

Nombre y apellidos:

---

1. [20 %] Se considera la función f en Python y se ejecuta 7 veces con los argumentos indicados. Escribir en la tabla el valor que devuelve y su tipo.

funcion.py

```
def f(x):
    if type(x)== int and x % 2 != 0:
        if x**2 <= 36:
            return "Los Peños"
        else:
            return x // 3
    elif type(x*3)== str:
         return x+'.'
    elif type(x) == float and x <= 0:
         return False
    elif type(x) == int and x >= 0:
              return x/2
    return "La Mina Fontoria"
```

Llamada Funcion	Valor Devuelto	Tipo de Valor
f(-12)		
f(-13)		
f(12)		
f('hola')		
f((1,2))		
f(5)		
f(-12.)		

2. [20 %] Una permutación es simplemente un nombre para reordenar. Así, las permutaciones de la cadena 'abc' son 'abc', 'acb', 'bac', 'bca', 'cab', y 'cba'. Observar que la propia cadena es una permutación de ella misma (la permutación trivial). En este ejercicio se pide escribir una función `testpermutacion` que tenga como entrada dos cadenas `s1` y `s2` y devuelva el booleano `True` si la cadena `s2` es una permutación de la cadena `s1` o el booleano `False` en caso contrario. Como requisito todos los caracteres de la cadena `s1` deben ser distintos.

```
>>> testpermutacion("acb","bca")
True
>>> testpermutacion("abc","ba2")
False
>>> testpermutacion("abc","baa")
False
```

3. [10 %] Escribe en notación hexadecimal la codificación UTF-8 de los caracteres cuyos *code points* en UCS son : U+132C, U+6B y U+71E.

4. [10 %] ¿Qué resultará de ejecutar las siguientes sentencias ?

```
>>> sol =[]
>>> sub = [1]
>>> sol.append(sub)
>>> sub.append(2)
>>> sol.append(sub)
>>> sub.remove(sub[-1])
>>> sol
```

<---------- Solucion

5. [10 %] ¿Qué resultará de ejecutar las siguientes sentencias ?

```
>>> A =[[0]*2]
>>> B = A*3
>>> B[1][0]=1
>>> B
```

<---------- Solucion

6. [15 %] Considera el siguiente fragmento de código:

```
1  def medias(L1, L2):
2      assert len(L1) >= len(L2)
3      medias = []
4      for i in range(len(L2)):
5          try:
6              medias.append(L1[i]/L2[i])
7          except ZeroDivisionError:
8              medias.append('infinito')
9          except:
10             raise ValueError('error en argumentos')
11     return medias
```

Escribe el valor de cada expresión, una vez ejecutado. Si se desencadena un error, escribe 'Error' :

```
medias([2,0,1],[1,2])  --------------->


medias([4.0,0,1],[2,3.1,0.0])------------------>


medias([2,0],['a',1,3])   ------------->


medias([1,2],[0,2 ])    -------------->


medias([1,2, 'hola'],[1,2,'hola'])  --------------->
```

7. [15 %] Considera el siguiente código.

```
roma_numeros = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
for k in roma_numeros:
      if roma_numeros[k] % 2 == 0:
            roma_numeros[k] *= 3
      else:
            roma_numeros[k] += 1
```

Escribe el valor de cada expresión, una vez ejecutado. Si se desencadena un error, escribe 'Error' :

```
print(len(roma_numeros))  ---------------->
```

```
print(roma_numeros['L'])    ---------------->
```

```
print(roma_numeros['M'])  ---------------->
```

```
print(100 in roma_numeros) ------------>
```

```
print(roma_numeros['X'])  ---------------->
```

```
print(roma_numeros['M'])  ---------------->
```