FUNDAMENTOS DE COMPUTACIÓN

Grado en ingeniería en tecnologías industriales

Universidad de Cantabria

Examen/Prueba (Aula)

4 de diciembre del 2018

Nombre y apellidos:

---

1. Supongamos que $B1, B2, B3, B4$ y $B5$ son variables booleanas inicializadas. Reescribe el siguiente código para que no contenga ningún `if` anidado. El código reescrito debe ser equivalente al código dado, es decir, debe devolver la misma salida independientemente del valor de la cinco variables booleanas inicializadas. *(1 punto)*

```
if B1:
    if B2:
        print('A')
    if B3:
        print('B')            SOLUCION------>
else:
    if B4 or B5:
        print('C')
    else:
        print('D')
```

---

2. **Célebre anécdota escolar de Gauß.**— Queremos calcular la suma $\sum_{i=1}^{n} i$. Escribe un programa que represente en el sistema unario y por duplicado la suma $\sum_{i=1}^{n} i$, dando un resultado similar al que sigue: *(1 punto)*

```
$ python suma2_1.py
Tope: 5
 1 +  5 = @ # # # # #
 2 +  4 = @ @ # # # #
 3 +  3 = @ @ @ # # #
 4 +  2 = @ @ @ @ # #
 5 +  1 = @ @ @ @ @ #
```

Esta suma duplicada toma la forma de un rectángulo de unidades (en el ejemplo, de dimensiones 6×5). Expresa su altura y su anchura en función de n y deduce una fórmula para $2\sum_{i=1}^{n} i$.

Aprovecha la fórmula para escribir una versión ágil del programa que suma los primeros n números.

---

3. Transformar a binario (usando a lo sumo 16 bits) el número decimal 237,12. *(1 punto)*

---

4. Escribir el siguiente código con bucles `while` en lugar de `for`: *(1 punto)*

```
suma = 0
for x in range(2, 8):
    if x % 2 == 0:
        for y in range(x):
            suma += y
print('suma:', suma)
```

¿Qué se muestra por pantalla, después de ejecutar el programa ? *(0,5 punto)*

---

5. ¿Cúal es el valor de las variables `list_A` `list_B` y `list_C` despúes de ejecutar el siguiente código ? *(1 punto)*

```
list_A = [17, 3]
list_A.append(5)                              list_A  ----->
list_B = list_A
list_B.append(21)            SOLUCIÓN         list_B  ----->
list_C = list(list_A)
list_A.append(8)
list_C.remove(3)                              list_C  ----->
```

---

6. Dado el siguiente diccionario: *(1 punto)*

```
mi_dic = {'uno' : 1, 2 : 2, 'cadena': 're', 'retener' : -100, 4 :16}
```

Escribe el valor de cada expresión. Si se desencadena un error, escribir 'Error' :

```
mi_dic['1']                              ---->

mi_dic[6-2]                              ---->

mi_dic[-100]                             ---->

mi_dic[mi_dic['cadena'] + 'tener']       ---->
```

---

7. Se considera los siguientes funciones: *(1,5 punto)*

```
def campoo(a):
    b = False
    for k in range(len(a)):
          b = b or cred(a ,k)
    return b

def cred(a,k):
    a[k] = a[k]-2
    return a[k] < 0
```

Escribir qué se muestra por pantalla cuando se ejecuta las siguientes instrucciones:

```
 >>> a = [-1,2,-3,-4]

 >>> print(liguerucu(a))       | ----->

>>> print(a)                   | ----->

>>> print(liguerucu(a))        | ----->

>>> print(a)                   | ----->
```

---

8. Escribe la función `desplazamiento` que tenga como argumento una tupla y devuelva la tupla que desplaza un lugar a la izquierda los elementos. *(1 punto)*

```
>>> desplazamiento(('a','b'))
('b', 'a')
>>> desplazamiento(('a', 'b', 'c','d'))
('b','c','d','a')
>>> desplazamiento((1,0,0,1,0,1))
(0,0,1,0,1,1)
```

---

9. Escribe un diagrama de flujo que represente el programa escrito abajo, a la izquierda, y el código Python correspondiente al diagrama de flujo de la derecha. ¿Tienen el mismo resultado final estos dos programas? En caso contrario, ¿en qué se diferencian? *(1 punto)*

```
cont = lista
n = len(cont)
i = 0
while i < 4 and i < n - 1:
    i += 1
print(cont[i])
```