**FUNDAMENTOS DE COMPUTACIÓN**

Grado en ingeniería en tecnologías industriales

Universidad de Cantabria

**Examen Final(Computador)**

1 de febrero del 2018

Nombre y apellidos:

---

1. ([25 %]) Diseña un programa que calcule y muestre el resultado de $f(x)$ para un valor $x$ de tipo FLOAT, que se pida por teclado, donde:

$$f(x) = \begin{cases} 0 & \text{si} \quad x \leq 1 \\ \sum_{n=1}^{m} \frac{x^n}{n!} & \text{si} \quad x > 1 \end{cases}$$

y donde $m$ es un entero introducido por el teclado y como mucho 14. Si m es mayor que 14, negativo o cero, volver a solicitarlo indefinidamente hasta que cumpla la condición dada. *(2 puntos)*

```
$ python ejercicio1.py
x?: -2
f(-2.0) = 0
```

```
$ python ejercicio1.py
x?: 2
Dame el número de sumandos 17
De 1 a 14
Dame el número de sumandos 0
De 1 a 14
Dame el número de sumandos 9
f(2.0) = 7.3887125220458545
```

2. ([35 %]) 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 recursiva y otra iterativa :

   1. `premutaciones_recursiva(cadena)`
   2. `premutaciones_iterativa(cadena)`

   que tenga como argumento una cadena y devuelva la lista de todas las permutaciones, no importa el orden, pero no se permiten repeticiones en la lista final.

```
>>> permutaciones_recursiva("abc")
['abc', 'acb', 'bac', 'bca', 'cab', 'cba']
>>> permutaciones_iterativa("G418")
['G418', 'G481', 'G148', 'G184', 'G841', 'G814', '4G18', '4G81', '41G8', '418G', '48G1', '481G',
'1G48', '1G84', '14G8', '148G', '18G4', '184G', '8G41', '8G14', '84G1', '841G', '81G4', '814G']')
>>>  permutaciones_recursiva('aeiou')
['aeiou', 'aeiuo', 'aeoiu', 'aeoui', 'aeuio', 'aeuoi', 'aieou', 'aieuo', 'aioeu', 'aioue', 'aiueo',
 'aiuoe', 'aoeiu', 'aoeui', 'aoieu', 'aoiue', 'aouei', 'aouie', 'aueio', 'aueoi', 'auieo', 'auioe',
 'auoei', 'auoie', 'eaiou', 'eaiuo', 'eaoiu', 'eaoui', 'eauio', 'eauoi', 'eiaou', 'eiauo', 'eioau',
 'eioua', 'eiuao', 'eiuoa', 'eoaiu', 'eoaui', 'eoiau', 'eoiua', 'eouai', 'eouia', 'euaio', 'euaoi',
 'euiao', 'euioa', 'euoai', 'euoia', 'iaeou', 'iaeuo', 'iaoeu', 'iaoue', 'iaueo', 'iauoe', 'ieaou',
 'ieauo', 'ieoau', 'ieoua', 'ieuao', 'ieuoa','ioaeu', 'ioaue', 'ioeau', 'ioeua', 'iouae', 'iouea',
 'iuaeo', 'iuaoe', 'iueao', 'iueoa', 'iuoae', 'iuoea', 'oaeiu', 'oaeui', 'oaieu', 'oaiue', 'oauei',
 'oauie', 'oeaiu', 'oeaui', 'oeiau', 'oeiua', 'oeuai', 'oeuia','oiaeu', 'oiaue', 'oieau', 'oieua',
 'oiuae', 'oiuea', 'ouaei', 'ouaie', 'oueai', 'oueia', 'ouiae', 'ouiea', 'uaeio', 'uaeoi', 'uaieo',
 'uaioe', 'uaoei', 'uaoie', 'ueaio', 'ueaoi', 'ueiao', 'ueioa', 'ueoai', 'ueoia', 'uiaeo', 'uiaoe',
 'uieao', 'uieoa', 'uioae', 'uioea', 'uoaei', 'uoaie', 'uoeai', 'uoeia', 'uoiae', 'uoiea']
```

3. ([40 %]) Este ejercicio está dedicado a la construcción de una clase denominada POLINOMIO, para hacer algunos cálculos con polinomios sobre los números reales, que tendrán dos atributos; por un lado, la lista de sus coeficientes, formadas por $n+1$ números $[a_0, a_1, \ldots, a_{n-1}, a_n]$ y, por otro, la variable `x` de tipo `str`.

   Un polinomio de grado $n$ en la variable $x$ es:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \longleftrightarrow [a_0, a_1, a_2 \ldots, a_{n-1}, a_n]$$

   Por ejemplo, el polinomio $p(x) = 3 - x + 5x^2 - x^4$ de grado 4 está identificado por la lista $[3, -1, 5, 0, -1]$ y la variable $x$.

   Se pide:

   - Crear la clase POLINOMIO y adaptar el método `__str__(self)` para mostrar por pantalla un polinomio de forma más agradable y no como una lista de números. Por ejemplo, el objeto

```
>>> P=Polinomio([3,-1,5,0,-1],'x')

>>> print(P)
3-x+5x**2-x**4
```

   - Implementar los métodos sumar, restar y multiplicar dos polinomios, adaptando estas operaciones con los métodos especiales:

```
__add__(self,otro)
 __sub__(self,otro)
__mul__(self,otro)
```

     Recordamos que dados dos polinomios $f(x)$ y $g(x)$ de grados $n$ y $m$ respectivamente:
     $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$ y $g(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_{m-1} x^{m-1} + b_m x^m$
     Supongamos que $n \geq m$ (similarmente si se tiene que $m \geq n$), entonces la suma y la resta es:

$$f(x) + g(x) = a_0 + b_0 + (a_1 + b_1)x + (a_2 + b_2)x^2 + \cdots + (a_m + b_m)x^m + a_{m+1}x^{m+1} + \cdots + a_n x^n$$

$$f(x) - g(x) = a_0 - b_0 + (a_1 - b_1)x + (a_2 - b_2)x^2 + \cdots + (a_m - b_m)x^m + a_{m+1}x^{m+1} + \cdots + a_n x^n$$

     Y el producto

$$f(x)g(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n+m-1} x^{n+m-1} + c_{n+m} x^{n+m} \quad \text{donde} \quad c_k = \sum_{i+j=k} a_i b_j$$

```
>>> P=Polinomio([3,-1,5,0,-1],'x')
>>> Q=Polinomio([0,-1,0,1,1],'x')
>>> R=Polinomio([1,-2],'x')
>>> print(P+Q)
3-2x+5x**2+x**3
>>> print(P-Q)
3+5x**2-x**3-2x**4
>>> print(Q*R)
-x+2x**2-2x**3+x**4-2x**5
```