Obligatorio

Ver condiciones de entrega del obligatorio en la página 3.

1 Álgebra de polinomios

El objetivo de este obligatorio es escribir una aplicación que nos permita realizar cálculos formales con polinomios con coeficientes racionales.

Para utilizar los números racionales alanza con hacer

```
from racionales import Racional
```

La expresión `Racional(2,4)` crea el racional $2/4 = 1/2$, el cual puede ser usado de la misma manera que un entero o real. Si se invoca la función con un sólo argumento: `Racional(3)`, se crea el racional $3/1$.

Inicialmente representaremos los polinomios como diccionarios, cuyas claves son los exponentes y sus valores los coeficientes. Por ejemplo, el polinomio $2X^{10} + \frac{1}{2}X^5 - 3$ puede representarse por el diccionario

```
p = {10:2, 5:Racional(1,2), 0:-3}
```

Los coeficientes que faltan se asumen iguales a cero. Podemos extraer el coeficiente que multiplica X^k con la expresión `p.get(k, 0)` (ver la ayuda para el método de diccionarios `{}.get`).

2 Tareas

1. Implementar las siguientes funciones sobre polinomios, utilizando la representación como diccionarios mencionada antes.
   (a) `sumar(p, q)`: debe devolver la suma de los polinomios `p` y `q`.
   (b) `multiplicar(p, q)`: debe devolver el producto de los polinomios `p` y `q`.
   (c) `potencia(p, n)`: debe devolver el polinomio `p` elevado a la `n`-ésima potencia, donde n es un entero no negativo.
   (d) `grado(p)`: debe devolver el grado del polinomio `p`. Convenimos que el grado del polinomio nulo es -1.
   (e) `dividir(p, q)`: debe devolver el polinomio cociente de dividir `p` entre `q`.

(f) `resto(p, q)`: debe devolver el resto de dividir `p` entre `q`. Las funciones anteriores deben satisfacer

```
p == sumar(multiplicar(q, dividir(p, q)), resto(p, q))
```

(g) `evaluar(p, x)`: debe devolver el racional que resulta de evaluar el polinomio `p` en el punto `x`. Por ejemplo, si `p` es el diccionario de más arriba, entonces

```
>>> polinomios.evaluar(p, -1)
-3/2
```

2. Escribir un nuevo tipo de objeto con nombre `Polinomio` que cumpla las siguientes condiciones

(a) Un objeto de tipo `Polinomio` se creará a partir de un diccionario como los mencionados más arriba. Es decir, debemos poder hacer

```
>>> p = polinomios.Polinomio({10:2, 5:1, 0:-3})
```

(b) Los objetos de tipo `Polinomio` deben imprimir en forma "legible", como en el siguiente ejemplo:

```
>>> polinomios.Polinomio({0:1, 2:Racional(-1,3), 5:4})
+ 4 X^5 - 1/3 X^2 + 1
```

Definir el método '`__repr__`' para que devuelva una cadena con una representación como la anterior.

(c) Los objetos de tipo `Polinomio` deben poder sumarse, restarse, multiplicarse, dividirse, etc. Más concretamente, definir los métodos adecuados para que se puedan realizar las siguientes operaciones:

i. `p + q` (métodos '`__add__(p,q)`' y '`__radd__(p,q)`')
ii. `p - q` (métodos '`__sub__(p,q)`' y '`__rsub__(p,q)`')
iii. `p * q` (métodos '`__mul__(p,q)`' y '`__rmul__(p,q)`')
iv. `p / q` (métodos '`__div__(p,q)`' y '`__rdiv__(p,q)`')
v. `p % q` (métodos '`__mod__(p,q)`' y '`__rmod__(p,q)`'; recordar que el operador `%` es el resto de dividir `p` entre `q`)
vi. `p ** n` (método '`__pow__(p,n)`')
vii. `p == q` (método '`__eq__(p,q)`')
viii. `p[i]` (coeficiente de X^i; método '`__getitem__(p,i)`')
ix. `p(x)` (evaluación de `p` en el racional `x`; método '`__call__(p,x)`')
x. `p.grado()` (devuelve el grado del polinomio `p`)

En las operaciones binarias anteriores y en la igualdad, se debe tener en cuenta que alguno de los operadores puede ser un objeto de tipo `Racional`, en cuyo caso se lo debe consierar como polinomio de grado 0.

3. Escribir las siguientes funciones sobre objetos de tipo `Polinomio`:

(a) `mcd(p, q)`: calcula el máximo común divisor de dos polinomios. (Sugerencia: recordar la función `mcd` del práctico 2.)

(b) `raices_racionales(p)`: en caso de que el polinomio `p` tenga todos sus coeficientes enteros, debe devolver la lista de raíces racionales de `p`. Si el polinomio tiene algún coeficiente no entero, se debe dar un error con la instrucción:

```
raise ValueError, 'el polinomio no tiene coeficientes enteros'
```

Recordar que si a/b es una raíz racional del polinomio con coeficientes enteros

$$a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0$$

entonces a divide a a_0 y b divide a a_n.

Sugerencias:

- Utilizar en todo lo posible las propiedades de polinomios definidas en la parte 2, ellas son suficientes para escribir las dos funciones anteriores.
- Para calcular el máximo común divisor de dos polinomios se puede usar el algoritmo de Euclides (que fue utilizado para enteros en el práctico 2).

3 Entrega del obligatorio

Este obligatorio debe ser realizado en grupos de no más de *tres* integrantes. Cada grupo debe enviar la solución por correo electrónico a la dirección `tornaria@cmat.edu.uy`.

La solución consistirá en un archivo con nombre `polinomios.py`. Dicho archivo debe implementar correctamente las funciones indicadas en la formulación anterior. Se puede verificar el funcionamiento del código bajando el archivo `verificar.py` de la página web del curso. El archivo `verificar.py` incluye una función con nombre `verificar()`, la cual importa el módulo `polinomios.py` y realiza varias comprobaciones automáticas. Es decir, el módulo `polinomios.py` que se entregue deberá pasar correctamente la prueba siguiente:

```
>>> import verificar
>>> verificar.verificar()
.............................
----------------------------------------------------------------------
Ran 32 tests in 0.200s

OK
.............................................
----------------------------------------------------------------------
Ran 50 tests in 0.167s

OK
```

La fecha límite de entrega es el:

Viernes, 20 de noviembre