MATLAB
REPRESENTACIÓN Y TRATAMIENTO DE POLINOMIOS.

Matlab nos brinda una serie de funciones para trabajar con los polinomios.
Aquí los polinomios se representan como vectores, por ejemplo, sea

$$\mathbf{p}(x) = 8x^4 - 5x^3 + x^2 + 3x + 4$$

en Matlab debemos ingresar a **p** de la siguiente manera
>> p = [8 - 5 1 3 4] **p = 8 - 5 1 3 4**

Observe que Matlab al responder no reescribe los corchetes. De ahora en adelante utilizaremos los términos vector y polinomio indistintamente para referirnos al mismo concepto.

Funciones básicas:

- **roots(p): Obtiene raíces de un polinomio dado**

Calcula las raíces de un polinomio cuyos coeficientes son los elementos del vector **p**.
Si **p** tiene N+1 componentes, el polinomio es **p**(1)*X^N + ... + **p**(N)*X + **p**(N+1).

>> q = [1 - 5 6] **q = 1 - 5 6**
>> roots(q) **ans = 3 2**

- **poly(v): Construye un polinomio a partir de sus raíces**

Retorna un vector (polinomio) cuyos elementos son los coeficientes del polinomio cuyas raíces son los elementos de **v**.
Puede apreciarse que **roots** y **poly** son funciones inversas.

>> v = roots(q) **v = 3 2**
>> q =poly(v) **q = 1 - 5 6**

O si desea verlo un poco más claro note que:
>> poly(roots(q)) **ans = 1 - 5 6**
>> roots(poly(v)) **ans = 3 2**

- **polyval(p, x): Evalúa un polinomio en un punto dado**

Si **p** tiene N+1 elementos, retorna el valor del polinomio al evaluarlo en x.
Es decir **y** = **p**(1)* $\mathbf{x}^N$ + **p**(2)*$\mathbf{x}^{N-1}$ + ... + **p**(N)***x*** + **p**(N+1)

- **conv(p, q): Multiplicación de polinomios**

conv viene de convolución, en el caso de los vectores convolucionar dos vectores es equivalente a multiplicarlos.
El resultado es un vector de longitud = longitud(**p**)+longitud(**q**)-1

Veamos un ejemplo:
Sean **p**(x) = 2x + 1 y **q**(x) = 3x + 4
>> p = [2 1] **p = 2 1**
>> q = [3 4] **q = 3 4**
>> r = conv(p, q) **r = 6 11 4**

Observe que retorna el polinomio **r**(x) = $6x^2 + 11x + 4$, y que la longitud del vector resultado es 3 = 2 + 2 -1.

- **deconv(p, q): División entre polinomios**

Se utiliza de la siguiente manera
>> [Q, R] = deconv(p, q);

El resultado de la división de **p** por **q** queda almacenado en la variable **Q**, y el resto de esta operación en la variable **R**.
Ejemplo:

```
>> r                r =   6   11    4
>> p                p =   2    1
>> [Q, R] = deconv(r, p);

>> Q                Q = 3   4
>> R                R = 0   0  0
```

Como era de esperarse (por que?).
Verifiquemos este resultado con la fórmula r = Q*p + R:

```
>> conv(Q, p) + R     ans = 6  11  4
>> r                        r = 6  11  4
```

Suma y resta:
Uno desearía poder sumar y restar polinomios de manera ágil y sencilla, lo óptimo sería utilizar los símbolos + y -, nada más.
Veamos que sucede:

```
>> p = [ 2  1  0  5 ];
>> q = [ 3  4 ];
>> p + q;
??? Error using ==> + Matrix dimensions must agree.
```

En castellano: Error usando ==> + Las matrices deben tener la misma dimensión.
Lo anterior nos informa que ocurrió un error al momento de utilizar el símbolo +, lo que sucede es que estos vectores (los vectores son un caso particular de matriz) no poseen la misma dimensión (cantidad de elementos).
La primer opción que tenemos para sortear este obstáculo es completar el vector p o q con tantos ceros a la izquierda como sean necesarios, de esta manera conseguimos un par de vectores que continúan representando los polinomios que teníamos inicialmente.

Nota: **q**(x) = 3x + 4 = $0x^3 + 0x^2 + 3x + 4 = 0x^n + 0x^{n-1} + ... + 0x^3 + 0x^2 + 3x + 4$
En nuestro caso:

```
>> p = [ 2  1  0  5 ];
>> q = [ 0  0  3  4 ];
>> p + q                    ans =   2    1    3    9
```

Otra posibilidad es crear una función que suma o resta correctamente, sin necesidad de que nosotros modifiquemos manualmente los vectores; si bien es una tarea más compleja, el esfuerzo se hará una única vez y luego siempre utilizaremos la función creada, además es una buena excusa para ponrese a

programar. La idea es que esta función haga el "relleno" con ceros apropiado y nos retorne la suma o resta.

Extenderemos en laboratorio...

Además de solucionar el problema planteado uno va familarizándose cada vez más con el entorno de programación, **una de las reglas básicas para aprender a programar es practicar.**

Factorización:

Supongamos que queremos factorizar un cierto polinomio **p** dado por la siguiente ecuación:

$$p(x) = (x^2 + 1)\,(x^7 - (1 + i)x^4 + ix)$$

Vamos a obtener todas las raíces de este polinomio.
La tarea de mostrar la factorización del polinomio, previo cálculo de sus raíces se puede efectuar mediante una función.
Comenzamos introduciendo el polinomio, observe que es un producto de dos polinomios y recuerde que el producto se realiza con la función **conv** y que los polinomios se representan con vectores.

$(x^2 + 1)$ → [1 0 1]
$(x^7 - (1 + i)x^4 + ix)$ → [1 0 0 -1-i 0 0 i]

Con la siguiente instrucción representamos a **p**.

```
>> p = conv([ 1  0  1 ], [ 1  0  0  -1-i  0  0  i  0 ]);
```

Ahora obtengamos sus raíces.

```
>> roots(p)     ans =  - 0.5000 - 0.8660i      0.0000 - 1.0000i      - 0.0000 - 1.0000i
                         1.0000 + 0.0000i      0.8660 + 0.5000i      - 0.8660 + 0.5000i
                         0.0000 + 1.0000i    - 0.5000 + 0.8660i                0
```

EJERCITACIÓN:

Resuelva mediante Matlab los ejercicios de la Práctica 2 y verique si los resultados que obtuvo son los correctos.

EJERCITACIÓN EN LABORATORIO:

Implemente las siguientes funciones:

- **Función que retorne el grado de un polinomio dado.**
- **Función que responda "Si" si el polinomio tiene como raíz a una dada.**

En laboratorio posiblemente pediremos que realice más funciones...