# Proba 1.

## Exercicio 1 (2 puntos)

O numero de DNI leva asociada unha letra, que pode ser calculada e se usa para verificar se hai algún dato erróneo no DNI, un cambio nun número fará que a letra non coincida.

O número de DNI está composto por 8 dixitos e unha letra. Para comprobar se está correcto hai que coller os 8 díxitos, interpretalos coma un número e calcular o resto da división do número entre 23. O que nos vai dar coma resultado un número entre 0 e 22. A asociación de letras é como se presenta na seguinte táboa.

| RESTO | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LETRA | T | R | W | A | G | M | Y | F | P | D | X | B | N | J | Z |

| RESTO | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|
| LETRA | S | Q | V | H | L | C | K | E |

Fai un programa que pida ao usuario un número de DNI coa letra e indique se é correcto ou non, para o que deberás calcular a letra que lle corresponde ao número e verificar que coincida coa letra proporcionada polo usuario.

- Deberás verificar qué o número está composto por 8 dixitos e unha letra, puidendo está estar en maiusculas ou minúsculas. Se non fose correcto indicarías que o formato non é o apropiado e pararárias o programa. (**0,5 puntos**)
- Se o formato é correcto deberás comprobar se a letra coincide co esperado, indicando finalmente cunha mensaxe se o número era correcto ou non. (**1 punto**)
- Deberás crear ao menos algúns test que chequee a función que crees. (**0,5 puntos**)

Algúns DNI con letra para que fagas probas: 65004204V, 30022846A

> Nos ficheiros que acompañan a proba tes unha lista de tuplas, onde o primeiro elemento e o resto, e o segundo e a letra maiúscula que lle toca.

# Exercicio 2 (6 puntos)

Cree unha clase Cifrador que reciba no constructor dous números enteiros e primos: $\mathbf{p}$ e $\mathbf{q}$

1. No construtor deberás verificar que son primos e proporcionar algún tipo de erro en caso de non selo. **(1 punto)**
2. No constructor deberás tamén xerar os seguintes atributos (calculados a partir de *p* e *q*): **(1 punto)**
   - **n**: que será o resultado de multiplicar *p* e *q*
   - **z**: que será o resultado de multiplicar (*p* - 1) e (*q* - 1)
3. Crea un novo atributo **k** que deberas elexir (mediante un bucle) que cumpla: **(1 punto)**
   - sexa maior que 1 e menor que **z**
   - o **máximo comun divisor** entre **z** e ese número **k** que estas a elexir sexa 1

   Para o **máximo comun divisor** hai unha función dentro do módulo **math** que se chama **gcd()**. Pasaslle dous números enteiros e che devolve o máximo comun divisor. O módulo **math** pertence a librería estándar.

4. Crea un novo atributo **j** que debe cumplir: **(1 punto)**
   - **j** debe ser un numero enteiro (int).
   - **j** debe ser igual a

   $$j = \frac{1 + x \times z}{k}$$

   , onde **x** é un numero enteiro que non coñecemos e que haberá que ir probando con 1, 2, 3, etc. ata atopar un **x** que facendo esa operación sexa un int. **k** e **z** son os calculados nos apartados anteriores. Cando probes un valor de **x** e queiras saber se **j** sería enteiro para ese valor de **x**, podes calcular o módulo e ver se da 0.

   $$1 + x \times z \mod k == 0$$

   (nese caso **j** seria un enteiro).
5. Crea unha nova clase chamada **Clave** que recibe dous parámetros no constructor, **n** e **y**. Os atributos da clase non poden ser cambiados unha vez que se lles pon valor no constructor. **(0,5 punto)**
   - Esta clase debe ter un método chamdo **procesar** que recive unha lista de int e devolve unha lista con cada elemento da lista orixinal transformada seguindo a seguinte operacion:

$$C = elemento^{y} \mod n$$

, sendo **elemento** cada un dos elementos da lista.

En python sería:

```
C = elemento ** y % n
```

6. Engada unha propiedade á clase *Cifrador* chama **clavePublica**, que devolva unha instancia de **Clave** cos seus atributos **n** e **k** (que obtes da instancia de Cifrador) **(0,25 punto)**

7. Engada unha propiedade á clase *Cifrador* chamada **clavePrivada**, que devolva unha instancia de **Clave** cos seus atributos **n** e **j** (que obtes da instancia de Cifrador) (**0,25 punto**)
8. Completa todo o que fagas con type hints e verifica que mypy da todo correcto. (**1 punto**)

Para que teñas de referencia e valores para testear:

Se creas unha instancia de Cifrador con **p** = 3 e **q** = 11, os valores que sarían son:

```
n = 33
z = 20
k = 3
j = 7
```

Isto debería funcionar:

```
if __name__ == "__main__":
    rsa = Cifrador(3, 11)
    publica = rsa.clavePublica
    privada = rsa.clavePrivada

    cadena = [14, 2, 4]
    cifrado = publica.procesar(cadena)
    descifrado = privada.procesar(cifrado)
    print(f"Orixinal: {cadena}")
    print(f"Cifrado: {cifrado}")
    print(f"Descifrado: {descifrado}")
    assert cadena == descifrado
```

## Exercicio 3 (2 puntos)

A cantidade de cruces por cero é unha bo xeito de tentar estimar a frecuencia dun sinal. Crea unha función (**cruces_cero**) que partindo dun array numpy nos devolva o numero de cruces por cero que hai. Se os valores do sinal se encontran dentro dun array (numpy) se podería facer:

1. Determinar o signo de cada un dos elementos, creando un novo array no que se poña un 1 se o número nesa posición era positivo e -1 se era negativo. (**0,5 puntos**) Ex: Partindo deste array:

```
[10, 10, 5, -3, 1]
```

Acabarías con este:

```
[ 1  1  1 -1  1]
```

2. Restar a este novo array (menos a ultima posición) o mesmo pero desprazado un posto. (**0,5 puntos**)

Ex:

A

```
[ 1  1  1 -1]
```

se lle restaría

```
[1  1 -1  1]
```

dando coma resultado:

```
[ 0  0  2 -2]
```

3. Contar o número de non ceros no array resultante. Poderías facer un novo array que conteña so os valores distintos de cero e ver o número de elementos que ten (size). **(0,5 puntos)**
4. Fai test (non fai falla que os fagas ao final, podes ilos facendo namentres fas os outros apartados) **(0,5 puntos)**