Creando fechas

∗ Escribe una clase `Date` que represente una fecha ∗

Implementa, *al menos*, los siguientes **métodos**:

```
def __init__(self, day: int, month: int, year: int):
```

- Crea los atributos: `day`, `month` y `year`
- Comprueba si la fecha es correcta → Entre el `1-1-1900` y el `31-12-2050`:
  - Si el día no es correcto se pondrá a 1.
  - Si el mes no es correcto se pondrá a 1.
  - Si el año no es correcto se podnrá a 1900.

```
def is_leap_year(year: int) -> bool:
```

- Método **estático**.
- Comprueba si el año dado es bisiesto.

```
def get_days_in_month(month: int, year: int) -> int:
```

- Método **estático**.
- Devuelve el número de días para el mes `month` del año `year`.

```
def get_days_in_month(month: int, year: int) -> int:
```

- Método **estático**.
- Devuelve el número de días para el mes `month` del año `year`.

```
def get_delta_days(self) -> int:
```

- Devuelve el número de **días transcurridos** desde el `1-1-1900` hasta la fecha del objeto `self`.

```
def weekday(self) -> int:
```

- Devuelve el **número de día de la semana** de la fecha `self`.
- Domingo(0), Lunes(1), Martes(2), Miércoles(3), Jueves(4), Viernes(5) y Sábado(6).

```
def is_weekend(self) -> bool:
```

- Indica si la fecha `self` cae en **fin de semana** (sábado o domingo).

```
def short_date(self) -> str:
```

- Devuelve la fecha `self` en formato `<day>/<month>/<year>`
- Día y mes con dos dígitos rellenando a cero si es necesario.
- Año con 4 dígitos.

`def __str__(self) -> str:`

- Devuelve la representación *string* de `self`.
- El formato será `'DOMINGO 11 DE AGOSTO DE 2024'`

`def __add__(self, days_to_add: int) -> Date:`

- Devuelve una nueva fecha sumando los días `days_to_add` a la fecha `self`.

`def __sub__(self, other: Date|int) -> Date|int:`

- Si `other` es de tipo `Date` se devolverá la **distancia en días** entre `self` y `other`.
- Si `other` es de tipo `int` se devolverá una **nueva fecha** → `self - other`.

`def __eq__(self, other: object) -> bool:`

- Indica si las fechas `self` y `other` son iguales.

`def __gt__(self, other: object) -> bool:`

- Indica si la fecha `self` es mayor (*más actual*) que `other`.

`def __lt__(self, other: object) -> bool:`

- Indica si la fecha `self` es menor (*más antigua*) que `other`.