Disponemos de un fichero CSV con información sobre recetas, cuyas líneas son de esta forma:

```
Tarta de Manzana,Postre,Baja,"manzanas, azúcar, harina, huevo, canela, mantequilla,
sal",60,200,2024-01-08,15.99
Spaghetti Bolognese,Plato principal,Media,"pasta, carne molida, tomate, cebolla, ajo",45,400,2024-
01-09,12.50
Ensalada César,Entrante,Baja,"lechuga, pollo, crutones, aderezo, parmesano",20,250,2024-01-10,8.99
```

La información de cada línea se corresponde con lo siguiente:

- **nombre**: nombre de la receta.
- **tipo**: tipo de receta (Postre, Plato principal, etc.).
- **dificultad**: dificultad de elaboración (Baja, Media, Alta).
- **Ingredientes**: ingredientes de la receta.
- **tiempo de preparación:** tiempo de elaboración en minutos.
- **calorías:** número de calorías de una porción.
- **fecha de creación:** fecha en la que se añadió la receta al dataset.
- **precio estimado:** precio de una porción en euros.

Así, la primera línea de los datos mostrados arriba se corresponde con la receta de una tarta de manzana de tipo postre, con baja dificultad de elaboración, con los ingredientes: manzanas, azúcar, harina, huevo, canela, mantequilla y sal, con un tiempo de elaboración de 60 minutos, 200 calorías, agregada al dataset el 8 de enero de 2024 y con un precio estimado por porción de 15,99 €.

NOTA: aquellas recetas cuya elaboración difiere bastante entre distintos países o territorios, no tienen lista de ingredientes. **En estos casos el valor de ingredientes en la tupla Receta debe ser None.** Vea como ejemplo la última receta que se muestra en la salida del test del Ejercicio 1.

Para almacenar los datos de una receta se usará **obligatoriamente** la siguiente NamedTuple:

```python
from typing import NamedTuple, Optional, List
from datetime import date
Receta = NamedTuple("Receta",
                    [("nombre", str),
                     ("tipo", str),
                     ("dificultad", str),
                     ("ingredientes", Optional[List[str]]),
                     ("tiempo_preparacion", int),
                     ("calorias", int),
                     ("fecha_creacion", date),
                     ("precio_estimado", float)
                    ])
```

Cree un módulo **recetas.py** e implemente en él las funciones que se piden a continuación, ajustándose a las cabeceras definidas con *typing* que se especifican en cada ejercicio. Cree también un módulo **recetas_test.py** y defina una función de test con los parámetros adecuados para cada función solicitada. Debe comprobar que la salida por consola de las funciones de test coincide con las mostradas en cada ejercicio, aunque en los casos de empate en ejercicios max/min, puede diferir (por ejemplo, si se pide la receta con más calorías, y existen varias con dicho número de calorías, el resultado podría ser cualquiera de ellas dependiendo de la implementación). Puede definir funciones auxiliares cuando lo considere necesario.

Para ser puntuado, es **imprescindible** que evite el código espagueti (ilegible, intrazable) y que siga la metodología y recomendaciones vistas en la asignatura. También es **obligatorio** el uso de los nombres de

datos en las variables y parámetros de tipo *Receta* (no puede usar los índices como si fueran tuplas sin nombre).

1. **lee_recetas**: dado el nombre y ruta de un archivo csv, devuelve una lista de tuplas de tipo `Receta` con los datos leídos del archivo. Defina las funciones auxiliares de parseo que crea convenientes. Le puede ser de ayuda la función `datetime.strptime(fecha_str, '%Y-%m-%d').date()` para el parseo de fechas. (1 punto)

   ```
   def lee_recetas(filename: str) -> List[Receta]
   ```

   ```
   === TEST EJERCICIO 1=======================================
   Recetas leídas: 100

   Mostrando las 2 primeras recetas:
           0-Receta(nombre='Tarta de Manzana', tipo='Postre', dificultad='Baja',
   ingredientes=['manzanas', 'azúcar', 'harina', 'huevo', 'canela', 'mantequilla', 'sal'],
   tiempo_preparacion=60, calorias=200, fecha_creacion=datetime.date(2024, 1, 8),
   precio_estimado=15.99)
           1-Receta(nombre='Spaghetti Bolognese', tipo='Plato principal', dificultad='Media',
   ingredientes=['pasta', 'carne molida', 'tomate', 'cebolla', 'ajo'], tiempo_preparacion=45,
   calorias=400, fecha_creacion=datetime.date(2024, 1, 9), precio_estimado=12.5)

   Mostrando las 2 últimas recetas:
           0-Receta(nombre='Sorbet de Mandarina', tipo='Postre', dificultad='Alta',
   ingredientes=['mandarinas', 'azúcar', 'clara de huevo', 'agua', 'licor de naranja'],
   tiempo_preparacion=25, calorias=180, fecha_creacion=datetime.date(2024, 4, 15),
   precio_estimado=18.25)
           1-Receta(nombre='Raviolis de Langosta', tipo='Plato principal', dificultad='Alta',
   ingredientes=None, tiempo_preparacion=50, calorias=400, fecha_creacion=datetime.date(2024, 4,
   16), precio_estimado=42.75)
   ```

2. **receta_mas_barata**: devuelve la receta más barata de entre las n recetas con menos calorías de alguno de los tipos de receta dados como parámetro. Si n no se especifica, se devuelve la más barata de todas las recetas de los tipos dados como parámetro. (1 punto)

   ```
   def receta_mas_barata(recetas: List[Receta],
                         tipos: Set[str],
                         n: Optional[int] = None) -> Receta
   ```

   ```
   === TEST EJERCICIO 2=======================================

   La receta más barata de las None con menos calorías de los tipos {'Postre', 'Entrante'} es:
           Receta(nombre='Gazpacho', tipo='Entrante', dificultad='Baja', ingredientes=None,
   tiempo_preparacion=25, calorias=120, fecha_creacion=datetime.date(2024, 2, 10),
   precio_estimado=6.95)

   La receta más barata de las 5 con menos calorías de los tipos {'Plato principal'} es:
           Receta(nombre='Lentejas Estofadas', tipo='Plato principal', dificultad='Baja',
   ingredientes=['lentejas', 'zanahoria', 'cebolla', 'tomate', 'ajo'], tiempo_preparacion=35,
   calorias=250, fecha_creacion=datetime.date(2024, 3, 5), precio_estimado=8.99)
   ```

3. **obten_ingredientes**: devuelve el conjunto de ingredientes necesarios para cocinar todas las recetas con ingredientes creadas en el intervalo de meses [mes1, mes2). Si mes1 no se especifica, se consideran todas las recetas creadas en un mes anterior a mes2. Si mes2 no se especifica, se consideran todas las recetas creadas en un mes posterior a mes1 o del mismo mes1. Si no se proporciona ninguno, no se tiene en cuenta el mes. (1 punto)

```
def obten_ingredientes(recetas: List[Receta],
                       mes1: Optional[int] = None,
                       mes2: Optional[int] = None) -> Set[str]
```

```
=== TEST EJERCICIO 3=====================================
Podrá comprobar mejor el resultado si ordena en el test el conjunto que devuelve la función
obtén_ingredientes

Hay 147 ingredientes de las recetas cuyos meses de creación están en el intervalo [None,None)
y son ['Grand Marnier', 'aceite de oliva', 'aceitunas', 'aderezo', ...]

Hay 109 ingredientes de las recetas cuyos meses de creación están en el intervalo [2,4) y son
['aceite de oliva', 'aceitunas', 'aderezo de limón', 'aguacate', 'ajo', 'albahaca',
'alcaparras', 'apio', 'arroz', ...]

Hay 137 ingredientes de las recetas cuyos meses de creación están en el intervalo [2,None) y
son
['Grand Marnier', 'aceite de oliva', 'aceitunas', 'aderezo de limón', 'agua', 'aguacate',
'ajo', 'albahaca', 'alcaparras', 'apio', ...]

Hay 91 ingredientes de las recetas cuyos meses de creación están en el intervalo [None,3) y
son
['aceite de oliva', 'aceitunas', 'aderezo', 'aderezo de limón', 'aguacate', 'ajo',
'albahaca', 'alcaparras', 'apio', 'arroz', 'arándanos', 'atún enlatado', 'avena', 'azafrán',
'azúcar', 'bechamel', 'berenjena', 'bizcochos', 'cacao', 'café', 'calabacín', 'calabaza',
'caldo de pollo', 'caldo de verduras',...]
```

4. **recetas_con_precio_menor_promedio**: devuelve una lista con n las tuplas (nombre, calorías) con menos calorías de las recetas cuyo precio sea menor que la media general (el promedio de todos los precios), ordenadas de menor a mayor número de calorías. (1,5 puntos)

```
def recetas_con_precio_menor_promedio(recetas: List[Receta], n: int)
                                      -> List[Tuple[str, int]]
```

```
=== TEST EJERCICIO 4=====================================
Observe que en el test las tuplas están precedidas de una numeración

Las 5 recetas con precio menor que el promedio son:
        0-('Ensalada de Frutas', 120)
        1-('Sopa de Tomate', 120)
        2-('Gazpacho', 120)
        3-('Brócoli al Vapor', 120)
        4-('Sopa de Calabaza', 150)
```

5. **receta_mas_ingredientes**: devuelve una tupla con el nombre y los ingredientes de la receta con mayor cantidad de ingredientes considerando solo recetas que contengan alguno de los ingredientes especificados en el parámetro. Si ingredientes no se especifica, se consideran todas las recetas. (1,5 puntos)

```
def receta_mas_ingredientes(recetas: List[Receta],
                                    ingredientes: Optional[Set[str]] = None)
                                    -> Tuple[str, List[str]]
```

```
=== TEST EJERCICIO 5=====================================

La receta con más ingredientes conteniendo alguno de estos: None es:
('Tarta de Manzana', ['manzanas', 'azúcar', 'harina', 'huevo', 'canela', 'mantequilla',
'sal'])
```

```
La receta con más ingredientes conteniendo alguno de estos: {'huevos', 'leche'} es:
('Mousse de Chocolate', ['chocolate', 'nata', 'azúcar', 'huevos', 'vainilla', 'limón'])
```

6. **ingredientes_mas_comunes_por_tipo**: devuelve un diccionario con los 3 ingredientes más comunes para cada tipo de receta. (2 puntos)

```
def ingredientes_mas_comunes_por_tipo(recetas: List[Receta])
                                        -> Dict[str, List[str]]
```

```
=== TEST EJERCICIO 6=======================================
Observe que en el test las tuplas están precedidas de una numeración

Los 3 ingredientes más comunes por tipo son:
        0-('Postre', ['azúcar', 'huevo', 'harina'])
        1-('Plato principal', ['cebolla', 'tomate', 'ajo'])
        2-('Entrante', ['tomate', 'cebolla', 'ajo'])
        3-('Acompañamiento', ['aceite de oliva', 'papas', 'sal'])
```

7. **mes_con_precio_medio_mas_alto**: devuelve el mes cuyo precio estimado promedio es el más alto, considerando solo los meses con al menos n recetas. (2 puntos)

```
def mes_con_precio_medio_mas_alto(recetas: List[Receta], n: int) -> int
```

```
=== TEST EJERCICIO 7=======================================

El mes con al menos 5 recetas con precio medio más alto es:
4

El mes con al menos 20 recetas con precio medio más alto es:
3
```