Ejercicio 1

```
def lee_registros_olimpiadas(filename: str) -> List [Registro]:
    registros = []
    with open(filename, 'r', encoding='utf-8') as file:
        csv_reader = csv.reader(file)
        next(csv_reader)
        for ciudad, fecha, pais, deporte, num_participantes, genero, medallas, sede
                in csv_reader:
            medallas = parsea_medallas(medallas)
            fecha = parsea_fecha(fecha)
            sede = parsea_boolean(sede)
            registro = Registro(ciudad.strip(), fecha, pais.strip(),
                deporte.strip(), int(num_participantes.strip()), genero.strip(),
                medallas, sede)
            registros.append(registro)
    return registros

def parsea_medallas(medallas_str: str) -> Medallas:
    oro, plata, bronce = medallas_str.split('-')
    oro, plata, bronce = int(oro.strip()), int(plata.strip()), int(bronce.strip())
    return Medallas(oro, plata, bronce)

def parsea_fecha(fecha_str: str) -> date:
    return datetime.strptime(fecha_str, '%Y-%m-%d').date()

def parsea_boolean(cadena: str) -> Optional[bool]:
    res = None
    cadena = cadena.upper()
    if cadena == 'SI':
        res = True
    elif cadena == 'NO':
         res = False
    return res
```

Ejercicio 2

```
def deportes_ambos_generos(registros: List[Registro], anyo: int) -> Set[str]:
    deportes_masc = set()
    deportes_fem = set()
    for r in registros:
        if r.fecha_inicio.year == anyo:
            if r.genero.upper() == "HOMBRE":
                deportes_masc.add(r.deporte)
            elif r.genero.upper() == "MUJER":
                deportes_fem.add(r.deporte)

    return deportes_masc.intersection(deportes_fem)
    # También: deportes_masc & deportes_fem
```

Alternativa:

```
def deportes_ambos_generos(registros: List[Registro], año: int) -> Set[Registro]:
    deportes_masc = set(r.deporte for r in registros
                    if r.fecha_inicio.year == año and r.genero.upper() == "HOMBRE")
    deportes_fem = set(r.deporte for r in registros
                    if r.fecha_inicio.year == año and r.genero.upper() == "MUJER")
    return deportes_masc.intersection(deportes_fem)
    # También: deportes_masc & deportes_fem
```

Ejercicio 3

```
def deportes_mas_frecuentes(registros: List[Registro], n: int, genero: str) ->
        List[Tuple[str, int]]:
    # Con Counter y most_common
    c = Counter(r.deporte for r in registros if genero == r.genero)
    return c.most_common(n)
```

Alternativa:

```
def deportes_mas_frecuentes(registros: List[Registro], n: int, genero: str) ->
        List[Tuple[str, int]]:
    # Con Counter y sorted + slicing
    c = Counter(r.deporte for r in registros if genero == r.genero)
    return sorted(c.items(), reverse=True)[:n]
```

Alternativa:

```
def deportes_mas_frecuentes(registros: List[Registro], n: int, genero: str) ->
        List[Tuple[str, int]]:
    # Con defaultdict y sorted + slicing
    c = defaultdict(int)
    for r in registros:
        if genero == r.genero:
            c[r.deporte] += 1
    return sorted(c.items(), key=lambda it:it[1], reverse=True)[:n]
```

Alternativa:

```
def deportes_mas_frecuentes(registros: List[Registro], n: int, genero: str) ->
        List[Tuple[str, int]]:
    # Con dict y sorted + slicing
    c = dict()
    for r in registros:
        if genero == r.genero:
            if r.deporte in c:
                c[r.deporte] += 1
            else:
                c[r.deporte] = 1
    return sorted(c.items(), key=lambda it:it[1], reverse=True)[:n]
```

Ejercicio 4

```
def deporte_con_mas_paises_distintos_con_oro(registros: List[Registro],
        genero: Optional[str] = None) -> str:

    paises_por_deporte = defaultdict(set)
    for registro in registros:
        if registro.medallas.oro > 0 and (genero is None or registro.genero == genero):
            paises_por_deporte[registro.deporte].add(registro.pais)

    return max(paises_por_deporte, key=lambda deporte:
        len(paises_por_deporte.get(deporte)))

    # También
    # item_max = max(paises_por_deporte.items(), key=lambda it:len(it[1]))
    # return item_max[0]
```

Ejercicio 5

```
def deportes_mas_participantes_de_genero_por_juego(registros: List[Registro],
        pais: str, genero: str) -> Dict[str, List[Tuple[str, int]]]:

    #1. diccionario que agrupa deportes y número de participantes por juego
    d = deportes_por_juego(registros, pais, genero)

    #2. Crear segundo diccionario a partir del anterior
    return {juego: sorted(lis_deportes, key=lambda t:t[1], reverse=True)[:3]
                for juego, lis_deportes in d.items()}

def deportes_por_juego(registros: List[Registro], pais: str, genero: str) ->
        DefaultDict[str, List[Tuple[str, int]]]:

    res = defaultdict(list)
    for r in registros:
        if r.pais == pais and r.genero == genero:
            juego = r.ciudad_olimpica + str(r.fecha_inicio.year)[-2:] # también [3:]
            res[juego].append((r.deporte, r.num_participantes))
    return res
```

Ejercicio 6

```
def deporte_con_todos_los_paises(registros: List[Registro]) -> bool:

    #1. Obtención de todos los paises
    paises = set(r.pais for r in registros)

    #2. Agrupación de paises por deporte
    d = paises_por_deporte(registros)

    #3. Esquema existe
    res = False
```

```
    for deporte, conj_paises in d.items():
        if paises == conj_paises:
            res = True
            break
    return res

def paises_por_deporte(registros: List[Registro]) -> DefaultDict[str, Set[str]]:
    res = defaultdict(set)
    for r in registros:
       res[r.deporte].add(r.pais)
    return res
```

Ejercicio 7

```
def anyo_con_mayor_incremento_participantes_de_pais(registros: List[Registro],
        pais: str) -> Tuple[int, int]:

    d = total_participantes_por_anyo_de_pais(registros, pais)
    juegos_ord = sorted(d.items(), key=lambda it:it[0])

    diferencias = ((j2[1]-j1[1], j2[0]) for j1, j2 in zip(juegos_ord, juegos_ord[1:]))
    return max(diferencias)

def total_participantes_por_anyo_de_pais(registros: List[Registro], pais: str) ->
        DefaultDict[Tuple[str, int], int]:

    d = defaultdict(int)
    for r in registros:
        if r.pais == pais:
            d[r.fecha_inicio.year] += r.num_participantes
    return d
```