# UNIDAD TEMÁTICA 3: Listas, Pilas, Colas

## PRACTICOS DOMICILIARIOS INDIVIDUALES #7

### Ejercicio #1

Desarrolla los algoritmos para implementar las operaciones de ***Union*** e ***Intersección*** sobre el **TDA LISTA**, utilizado para representar un **CONJUNTO**. Los conjuntos de entrada se encuentran ordenados.
Debes seguir las normas para desarrollo de algoritmos en seudocódigo brindadas en el curso:

- especificación en lenguaje natural,
- pre y post condiciones,
- seudocódigo y
- especificación de casos de prueba.
- Analiza detalladamente el orden del tiempo de ejecución de estos algoritmos.

### Ejercicio #2

Partiendo de las implementaciones que tienes del TDA Lista, desarrolla un TDA Conjunto que, en particular, implemente las funciones:

- **Union**: dada una segunda instancia del TDA Conjunto, devuelve una tercera, producto de aplicar esta operación sobre los elementos propios y de la segunda.
- **Interseccion**: dada una segunda instancia del TDA Conjunto, devuelve una tercera, producto de aplicar esta operación sobre los elementos propios y de la segunda.

Una **posible** interfaz para el TDA CONJUNTO sería:

```
public interface IConjunto<T> extends ILista<T> {
   public IConjunto<T> union(IConjunto<T> otroConjunto);
   public IConjunto<T> interseccion(IConjunto<T> otroConjunto);
   }
```

### Ejercicio #3

1. Crea dos instancias del TDA Conjunto, genérico en TAlumno, para representar los cursos “Algoritmos y Estructuras de Datos I – AED1” y “Programación Funcional – PF”.
2. Crea varias instancias de un **TAlumno** (con cédula – sólo 4 dígitos -, Nombre y Apellido)
3. Asigna (inserta) algunos alumnos en el primer curso y a otros en el segundo. Verifica que algunos alumnos estén presentes en ambos cursos.
4. Emite el listado de alumnos que se encuentran matriculados en cualquiera de los dos cursos o en ambos (utilizando el método de **union)**
5. Emite el listado de alumnos que se encuentran matriculados *simultáneamente* en ambos cursos (utilizando el método de **interseccion)**
6. Crea casos de prueba en JUnit de acuerdo a lo desarrollado en el Ejercicio 1 y verifica que los métodos funcionan como esperado.

**NOTA IMPORTANTE: los conjuntos no pueden contener elementos duplicados**