# Práctica de Recursividad de Pila con Listas en C#

Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Algoritmos y Estructuras de Datos I

**Profesores**: M.Sc. Jeff Schmidt Peralta y M.Sc. Jason Leitón Jiménez y M.Sc. Leonardo Araya Martinez
**Estudiante**: Steven Aguilar Alvarez

## Introducción

La recursividad de pila aplicada a listas representa una técnica fundamental en programación que permite manipular estructuras de datos de manera elegante y concisa. Esta práctica explora diferentes estrategias para procesar listas utilizando llamadas recursivas.

## 1. Ejercicios de Programación

1. **División por Umbral** Escriba un método recursivo SplitListByThreshold(threshold, list) que reciba un dígito y una lista y obtenga dos listas, la primera compuesta por los dígitos mayores o iguales al dígito dado y la segunda compuesta por los dígitos menores al dígito dado:
   - SplitListByThreshold(5,[1,0,2,9,0,9,9]) → ([9,9,9], [1,0,2,0])
   - SplitListByThreshold(8,[1,0,2,6,0]) → ([], [1,0,2,6,0])
2. **Calificación de Competencia** Escriba un método recursivo CalculateScore(scores) que reciba una lista con calificaciones de 10 jueces (escala 1-100), elimine la más alta y la más baja y retorne el promedio de las restantes:
   - CalculateScore([6,8,8,8,8,8,8,10,8,8]) → 8
3. **Verificación de Intercambio** Escriba un método recursivo IsExchanged(number) que reciba un número entero e indique si cada uno de los dígitos que lo componen se intercambian entre pares e impares:
   - IsExchanged(943652) → true
   - IsExchanged(3423830) → false
4. **Conteo de Divisibilidad** Escriba un método recursivo CountDivisibleDigits(divisor, number) que reciba un dígito y un número entero y obtenga dos contadores, divididos exactamente y no divididos:
   - CountDivisibleDigits(2,1029099) → (1, 6)
   - CountDivisibleDigits(3,396366) → (6, 0)
5. **Conteo de Parejas** Escriba un método recursivo CountPairs(number) que reciba un número entero y cuente las veces que aparecen dígitos en parejas:
   - CountPairs(41234426601) → 2
   - CountPairs(333) → 1

## Observaciones Finales

Los ejercicios deben implementarse utilizando exclusivamente recursividad de pila en C#. Es fundamental mantener un control adecuado de la pila de ejecución mediante el uso apropiado de casos base y llamadas recursivas.