**ALGORITMOS Y ESTRUCTURA DE DATOS**

**Módulos cortos tipo parte 2do Parcial**

1. (1,5 pts) Defina en C/C++ la estructura de datos que un nodo de productos con **Info**

**CodP** (Código de Producto – entero) **PU** (Precio Unitario, decimal) **TUV** (Total de Unidades Vendidas, entero) **TotF** (Total Facturado, decimal)

2. (1,5 pts) Escriba en C/C++ la cabecera de un módulo **AcumularFactProd** que recibe ***una lista de productos, un cód de producto y la cantidad vendida de una venta, y actualiza la lista de productos***.

3. (7 pts) Diseñe el algoritmo de dicho módulo, que busca dicho producto en la lista.

En caso de encontrarlo actualiza los campos **TUV** y **TotF** del nodo a partir de los datos de la venta.

Y en caso de no encontrarlo, solicita el ingreso del **PU** del producto, e inserta un nodo con los datos de ese producto al final de la lista.

Un poquito más complejo (con un vector dentro del nodo)

1. (1,5 pts) Defina en C/C++ la estructura de datos que un nodo de productos con **Info**

**CodP** (Código de Producto – entero) **PU** (Precio Unitario, decimal) **VSuc** (vector de 5 sucursales conteniendo en cada posición: **TUV** (Total de Unidades Vendidas, entero) y **TotF** (Total Facturado, decimal))

2. (1,5 pts) Escriba en C/C++ la cabecera de un módulo **AcumularFactProd** que recibe ***una lista de productos, un cód de producto, una sucursal (1 a 5) y la cantidad vendida de una venta, y actualiza la lista de productos***.

3. (7 pts) Diseñe el algoritmo de dicho módulo, que busca dicho producto en la lista.

En caso de encontrarlo actualiza los campos **TUV** y **TotF** (de la sucursal indicada) en el nodo, a partir de los datos de la venta.

Y en caso de no encontrarlo, solicita el ingreso del **PU** del producto y la sucursal de la venta, e inserta un nodo con los datos de ese producto al final de la lista (e inicializa el resto de las sucursales de ese producto con sus datos en 0).