UTN – FRBA – Algoritmos y Estructuras de Datos – Segundo Parcial

Apellido y Nombre: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Legajo: \_\_\_\_\_\_\_\_\_

Email (en MAYUSC): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Temas evaluados: Resolución de problemas, abstracción de datos y lenguaje de programación. En particular en puntos 1 y 2: Estructuras enlazadas, y en punto 3: Estructuras estáticas*

**1 - Implementación de listas doblemente enlazadas**

*Problema -* Un centro médico lleva las carátulas de las historias clínicas de sus pacientes en una lista doblemente enlazada, que contiene como Info por cada paciente

**Leg** - Legajo del paciente de tipo entero / **AN** - Apellido y Nombre de tipo string / **NR** - Nro de Registro (de la Base de Datos de Historias Clínicas) de tipo entero

1a. Declarar en C++ la estructura de datos del nodo

1b. desarrollar en C++ o algoritmo (la cabecera debe ser en C++) la función **UbicarPaciente**, que recibe un puntero no nulo a la lista de pacientes, un Nro de Legajo a ubicar, y **retorna** la dirección del nodo de dicho paciente, o nulo si no se encuentra. El mecanismo de búsqueda ya optimizado para este conjunto debe ser el siguiente: primero busca desde el puntero recibido hacia el final de la lista, y en caso de no encontrarlo, busca desde el puntero recibido hacia el comienzo de la lista, donde nuevamente puede estar o no.

1c. Desarrollar C++ o algoritmo la parte del programa principal que sin necesidad de inicializar las variables/parámetros que utilice, invoque a **UbicarPaciente**, y en caso de no encontrarlo solicita AN y NR y lo agregue al final de la lista doblemente enlazada utilizando y actualizando un puntero a Fin que contiene la lista (lista no vacía).

1d. (sólo para quien debe el primer parcial) Obtener la cantidad de pacientes con Nro de Registro entre 600 y 800, y el Apellido y Nombre del Paciente de mayor Legajo.

**2 - Implementación de colas**

*Problema –* Caja rápida de un Supermercado

2a. Desarrollar en C++ o algoritmo (la cabecera debe ser en C++) la función **EncolarEnCajaRapida** que recibe punteros **Fte** y **Fin**, y **CA** que es la cantidad de artículos en un carrito (de tipo entero), y agrega el carrito a la cola sólo si la cantidad de artículos en el mismo es menor o igual a 10 unidades. La info del nodo contiene solamente la cantidad de unidades del carrito. Debe **retornar** a su vez Verdadero o Falso según haya agregado el carrito o no a la cola.

**3 – Para mayor puntaje - Armar conjunto sin repetición**

*Problema –* Desarrollar en C++ o algoritmo la función **BuscaAgrega** que recibe un vector que contiene Códs de producto, recibe un cód de producto a buscar y un tope, y debe agregar el producto al vector actualizando tope sólo si el producto aún no estaba. Retorna la posición donde lo encontró o agregó.