Trabajo Práctico III

Ejercicio 1

Dada una lista de elementos enteros, simplemente enlazada implementar el método de `Lista* Intersección (Lista &lista)` que devuelve un puntero a una lista que es la intersección entre la lista que invoca al método y la que se pasa por parámetro,

Para este ejercicio se consideran dos versiones en donde las listas están:

Desordenadas `interseccion_A`

En este caso recorro ambas listas verificando si cada elemento se encuentra en la otra lista. La desventaja de esta versión es que para grandes cantidades puede tardar demasiado y es ineficiente

Ordenadas `interseccion_B`

En esta versión, se guardan dos valores `i, j` que sirven de cursores apuntando uno a cada lista. Siempre que el elemento al que apunto el cursor sea menor al otro, se aumenta el primero y en caso de ser iguales se aumentan ambos cursores a la siguiente posición.

Ejercicio 2

La clase Lista Circular tiene un tamaño máximo guardado en una constante MÁXIMO. La lista puede crecer hasta esa cantidad de elementos, una vez alcanzado el máximo se reemplaza el elemento más antiguo por el nuevo elemento.

También se implementaron los métodos `mas_antiguo()` que devuelve el elemento más antiguo de la lista y `promedio()` que devuelve el promedio entre todos los elementos.

Ejercicio 3

En este ejercicio se crean las clases Carrera y Buscador Carreras además de Lista y Nodo.

Carrera representa a una carrera universitaria con una Lista de materias, duración y nombre.

Por otro lado, Buscador Carreras tiene dos métodos `sugerir_carreras()` y `materias_en_comun()` . Con el primero se retorna una nueva Lista con carreras sugeridas en base a las `materias predilectas` y la duración de todas las carreras. Su otro método devuelve la cantidad de materias predilectas que se encuentran en cada carrera

En caso de que la cantidad de materias predilectas sea mayor que 3 y la duración de una carrera sea menor o igual a una duración máxima, la carrera en cuestión es sugerible.