División de Ingeniería
Departamento de Tecnologías de Información y Electrónica
Estructuras de Datos (TC1018)
Profesor: Dr. Vicente Cubells Nonell

Tecnológico de Monterrey

Tarea 3. Listas enlazadas

Orientaciones sobre la entrega de la tarea

La respuesta a cada ejercicio es un programa funcionando que modele el problema planteado. Como respuesta de la tarea debe entregarse el código fuente de cada ejercicio por separado (cada ejercicio es un proyecto diferente). Para la evaluación, el profesor compilará el proyecto de cada ejercicio, el cual debe funcionar sin errores y analizará el código fuente.

La respuesta de la tarea debe subirse a GitHub y en BB debe subirse la liga de acceso al repositorio. La tarea estará activa en BB hasta el 6 de marzo de 2015 a las 23:55 horas.

No se aceptan trabajos fuera de fecha ni por correo, en ambos casos la calificación de la tarea será 0 puntos.

Ejercicio 1.

Haga un programa que genere y almacene en una lista enlazada `n` números primos que se encuentren en el rango `[p..q]` y después los muestre en forma inversa. Los valores de `n`, `p` y `q` deben ser definidos por el usuario.

Ejercicio 2.

Programe una aplicación que permita ingresar `n` números reales a una lista enlazada y `m` números reales a otra lista, siendo `m` y `n` elegibles por el usuario (entrados por consola). A las listas anteriores les llamaremos `N` y `M` respectivamente.

Después de tener ambas listas, la aplicación debe ser capaz de realizar las siguientes operaciones:

- `N ∪ M` (Los elementos duplicados deben aparecer una sola vez)
- `N - M`
- `M - N`
- `N * M`
- `N ∩ M`

Los resultados de la operación seleccionada por el usuario deben almacenarse en otra lista enlazada `P` y mostrarse por pantalla junto con la información de las listas `N` y `M`.

Por ejemplo:

```
Lista N: (1, 2, 3, 4)

Lista M: (5, 6, 7, 4)
```

Si el usuario selecciona `N ∪ M`

```
Lista P: (1, 2, 3, 4, 5, 6, 7)
```

Si el usuario selecciona `N - M`

```
Lista P: (1, 2, 3)
```

Si el usuario selecciona `M - N`

```
Lista P: (5, 6, 7)
```

Si el usuario selecciona `N ∩ M`

```
Lista P: (4)
```

Si el usuario selecciona `N * M`

```
Lista P: (5, 6, 7, 4, 10, 12, 14, 8, 15, 18, 21, 12, 20, 24, 28, 16)
```

Ejercicio 3.

Realice un programa que permita a un usuario entrar los datos de *n* personas y almacenarlos en una lista enlazada ordenada, de tal manera que en todo momento las personas se encuentren ordenadas alfabéticamente por el apellido y luego por el nombre. El usuario puede entrar tantas personas como desee. La aplicación debe permitir en todo momento:

- Añadir una persona
- Buscar una persona
- Borrar una persona
- Visualizar los datos de todas las personas almacenadas
- Borrar todas las personas
- Determinar la cantidad de personas existentes

Para resolver el ejercicio debe, a partir de la clase `LinkedList`, crear una nueva clase denominada `OrderedLinkedList` que herede de la clase anterior y sobreescriba y/o defina los métodos necesarios para mantener la lista ordenada en todo momento. Esta clase debe ser genérica.

Los datos de las personas que interesan son: `nombre`, `apellidos`, `edad` y `fecha de nacimiento`.