Estructuras de Datos
Examen Final
Agosto - diciembre de 2012

- Tiene dos horas para resolver el examen.
- Abra el proyecto FinalEDdiciembre2012 que se encuentra en ComunidadITAM en el archivo con el mismo nombre .rar y descomprímalo. Agregue las clases y los métodos necesarios de acuerdo a las preguntas del examen.
- **No puede modificar ninguna parte del código que se le va a proporcionar** excepto para agregarle los elementos que se solicitan en las siguientes preguntas. La entrega final debe consistir de un solo proyecto de NetBeans comprimido que contenga dos programas ejecutables (es decir, dos clases que tengan un método *main*), uno por cada pregunta, así como las demás clases e interfaces necesarias para correr dichos programas.
- Asegúrese que en todas las clases creadas o modificadas aparezca su nombre.
- **Se calificará la eficiencia en todos los problemas**.
- **Todas las soluciones planteadas deben ser generales (para cualquier entrada). Los casos de prueba que se dan en los *main* son sólo eso: casos de pruebas.**

1. [4 PUNTOS] Escriba un método estático con la firma que aparece más abajo. El método recibe tres conjuntos y debe regresar *true* si el primero de ellos (cI) es el conjunto intersección de los otros dos (cA y cB). En caso contrario debe regresar *false*. **La solución debe ser recursiva.** Debe probar la solución con el código que se le proporciona en el método *main* de la clase Main.

```
public static <T> boolean esIntersección(ConjuntoADT<T> cI,
ConjuntoADT<T> cA, ConjuntoADT<T> cB)
```

2. [6 PUNTOS] Esta pregunta tiene que ver con listas **ordenadas crecientemente** implementadas usando estructuras doblemente enlazadas. Se le proporciona un método *main* en la clase Main2, así como varias interfaces y clases, algunas sólo parcialmente definidas, necesarias para que este método funcione. Para resolver este problema debe implementar todos los métodos faltantes de la clase Main2 y de las clases ListaDoblementeEnlazada<T> y ListaOrdenadaDoblementeEnlazada<T>. Los métodos de estas últimas dos clases que no son necesarios para resolver el problema pueden implementarse (para que el compilador no marque error) de tal forma que simplemente lancen una excepción de tipo UnsupportedOperationException. Nótese que, por la manera en que están implementados los métodos *compareTo* de las clases Circle y Rectangle, son las áreas de dichas figuras geométricas las que se van a comparar para decidir el orden relativo entre instancias de las clases.