1. Cree una clase **Counted** que contenga un entero **id** y un entero estático **count**. El constructor debería empezar: **Counted( ) : id(count++) {.** Debería también imprimir su **id** y que está siendo creado. El destructor debería imprimir que se está destruyendo. Pruebe la clase.
2. Compruebe que **new** y **delete** siempre llaman a los constructores y destructores sobre instancias de **Counted**. Pruebe también creando arreglos de estos objetos sobre el heap.
3. Cree un **vector<Counted*>** y llénelo con punteros a instancias de **Counted**. Recorra el vector imprimiendo los objetos y recórralo nuevamente llamando a **delete** sobre cada uno.
4. Cree dinámicamente un arreglo de instancias de **Counted**. Llame a **delete** para el puntero resultante sin usar corchetes. Explique el resultado.
5. Cree un objeto de la clase **Counted** usando **new**, castee el puntero resultante a un **void*** y llame a **delete** sobre él. Explique el resultado.
6. Implemente una lógica para calcular la cantidad de memoria que puede utilizar del heap (utilice memory exhaustion).
7. Cree una clase con los operadores new y delete sobrecargados, ambos simples y version arreglo. Demuestre que ambas versiones funcionan.
8. Suponga que está diseñando una clase que será instanciada múltiples veces, pero que por diseño, nunca habrá más de N instancias simultaneas. Sobrecargue los operadores **new** y **delete** para esta clase para que utilicen un mecanismo eficiente de alocación (sin fragmentación y con costo constante).