Universidad de Costa Rica	**Examen 02**	CI-1201 Programación II - 2013a
Escuela de Computación	15-Jun-2013	Profesor Jeisson Hidalgo-Céspedes

En cada ejercicio se evaluará la eficiencia del código y las buenas prácticas de programación como: el uso de identificadores significativos, la indentación, escritura correcta de llaves {} y el uso adecuado de la palabra reservada const. Se dispone de tres horas para entregar la prueba y debe realizarse en forma estrictamente individual.

Escriba en C++ una plantilla para generar clases `Conjunto` (en inglés `Set`), las cuales representan conjuntos finitos de valores de algún tipo de datos. Note que por definición un conjunto matemático no permite valores repetidos. Dadas las limitaciones de los compiladores sobre las plantillas, puede escribir su clase completa en el archivo de encabezado (`.h`). Debe implementar al menos los siguientes miembros.

1. [5%] **Miembros de datos**: Debe almacenar los valores en alguna estructura de datos que pueda crecer dinámicamente. No puede utilizar (instanciar o heredar de) clases contenedoras de la biblioteca estándar de C++ ni hechas durante el curso. Debe además almacenar otros valores que ayuden a saber el tamaño o la capacidad del contenedor. Recuerde que no se conoce de antemano el tipo de dato de los valores.
2. [5%] **Constructor por defecto**: Crea un conjunto vacío.
3. [5%] **`Set(n)`**: Crea un conjunto con capacidad para `n` valores. Note que `n` es una indicación de capacidad y no se trata de convertir `n` en un conjunto.
4. [15%] **Sobrecargas fundamentales para manejo de memoria dinámica**: Dado que la clase `Set` utiliza memoria dinámica, debe evitar fugas de memoria o accesos inválidos a toda costa. Es importante que estos métodos sean públicos e implementados, no se deben declarar como privados.
5. [5%] **Método `count()`** retorna la cantidad de valores contenidos en el conjunto. Método **empty()** retorna `true` si el conjunto está vacío.
6. [5%] **Método `has(v)`**: Retorna `true` si el valor `v` se encuentra en el conjunto, `false` en caso contrario. ¿Es su implementación eficiente? Justifique. Si la respuesta es negativa, indique cómo hacerlo eficiente y la consecuencia en los otros métodos de la clase.
7. [10%] **Método `add(v)`**: Agrega una copia del valor `v` en el conjunto. Retorna verdadero si el valor pudo ser agregado, falso si no hay memoria suficiente.
8. [10%] **Operador +**: Implementa la unión del conjunto con algún otro. La unión de dos conjuntos A ∪ B es el conjunto de elementos que están en A, en B o en ambos. **Operador +=** almacena en el conjunto que está en el lado izquierdo del operador, el resultado de la unión de ese conjunto con el que aparece a la derecha. No invoque el operador de asignación en la implementación del operador `+=`.
9. [10%] **Operador ***: Implementa la intersección del conjunto con alguno otro. La intersección de dos conjuntos A ∩ B es el cojunto de elementos que están tanto en A como en B. **Operador *=** almacena en el conjunto que está en el lado izquierdo del operador, el resultado de la intersección de ese conjunto con el que aparece a la derecha. No invoque el operador de asignación en la implementación del operador `*=`.
10. [5%] **Operador ==**: Retorna `true` si el conjunto que está al lado izquierdo del operador tiene exactamente los mismos elementos que el conjunto que está al lado derecho del operador.
11. [10%] **operator >>**: permite leer un conjunto a partir de un archivo. No debe imprimir nada en la salida estándar, sólo leer los valores uno tras otro del archivo y almacenarlos en el conjunto hasta encontrar el final del archivo. **Operador <<**: imprime los valores del conjunto a un archivo separando los valores por comas.
12. [15%] **Clase `ConstIterator`**: Implemente una clase `ConstIterator` que permita tener acceso a los elementos del conjunto. Su iterador debe ser capaz de ejecutar el código que se encuentra en el `main()` de abajo. Implemente métodos en la clase `Set` para obtener un iterador que inicie el recorrido por el contenedor, y otro que indique una posición no válida o de finalización del contenedor.

Recuerde implementar en línea los métodos que son candidatos a ello. Su clase `Set` debe hacer funcionar el siguiente `main()`:

```
int main()
{
   Set<long> set1;
   cout << "Input first set (EOF to finish):\n";
   cin >> set1;

   Set<long> set2;
   cout << "Input second set (EOF to finish):\n";
   cin.clear();
   cin >> set2;

   Set<long> set3 = set1 + set2;

   cout << set1 << " + " << set2 << " = " << set3 << endl;
   cout << set1 << " * " << set2 << " = " << set1 * set2 << endl;

   Set<long> set4;
   for ( Set<long>::ConstIterator itr = set3.begin(); itr != set3.end(); itr++ )
      set4.add( *itr );

   cout << set3 << " == " << set4 << ": " << set3 == set4 << endl;

   return 0;
}
```