set.cm

```
/*
    ========================================================================

    Copyright (c) 2012-2016 Seppo Laakko
    http://sourceforge.net/projects/cmajor/

    Distributed under the GNU General Public License, version 3 (GPLv3).
    (See accompanying LICENSE.txt or http://www.gnu.org/licenses/gpl.html
        )

    ========================================================================
    */

// Copyright (c) 1994
// Hewlett-Packard Company
// Copyright (c) 1996
// Silicon Graphics Computer Systems, Inc.
// Copyright (c) 2009 Alexander Stepanov and Paul McJones

using System;
using System.Concepts;

namespace System.Collections
{
    public class Set<T, C = Less<T>> where T is Semiregular and C is
        Relation and C.Domain is T
    {
        public typedef T ValueType;
        public typedef T KeyType;
        public typedef C Compare;
        private typedef Set<ValueType, Compare> Self;
        private typedef RedBlackTree<KeyType, ValueType, Identity<
            ValueType>, Compare> TreeType;
        public typedef TreeType.ConstIterator ConstIterator;
        public typedef TreeType.Iterator Iterator;

        public nothrow inline Iterator Begin()
        {
            return tree.Begin();
        }
        public nothrow inline ConstIterator Begin() const
        {
            return tree.CBegin();
        }
        public nothrow inline ConstIterator CBegin() const
        {
            return tree.CBegin();
        }
        public nothrow inline Iterator End()
        {
            return tree.End();
```

```
        }
        public nothrow inline ConstIterator End() const
        {
            return tree.CEnd();
        }
        public nothrow inline ConstIterator CEnd() const
        {
            return tree.CEnd();
        }
        public nothrow inline int Count() const
        {
            return tree.Count();
        }
        public nothrow inline bool IsEmpty() const
        {
            return tree.IsEmpty();
        }
        public nothrow void Clear()
        {
            tree.Clear();
        }
        public nothrow inline Iterator Find(const KeyType& key)
        {
            return tree.Find(key);
        }
        public nothrow inline ConstIterator Find(const KeyType& key)
           const
        {
            return tree.CFind(key);
        }
        public nothrow inline ConstIterator CFind(const KeyType& key)
           const
        {
            return tree.CFind(key);
        }
        public inline Pair<Iterator, bool> Insert(const ValueType& value)
            where T is Copyable
        {
            return tree.Insert(value);
        }
        public nothrow inline bool Remove(const KeyType& key)
        {
            return tree.Remove(key);
        }
        public nothrow inline void Remove(Iterator pos)
        {
            tree.Remove(pos);
        }
        private TreeType tree;
    }

    public nothrow inline bool operator==<T, C>(const Set<T, C>& left,
       const Set<T, C>& right) where T is Regular and C is Relation and C
```

```
        .Domain is T
    {
        return left.Count() == right.Count() && Equal(left.CBegin(), left
            .CEnd(), right.CBegin(), right.CEnd(), EqualTo<T>());
    }

    public nothrow inline bool operator<<T, C>(const Set<T, C>& left,
        const Set<T, C>& right) where T is Semiregular and C is Relation
        and C.Domain is T
    {
        return LexicographicalCompare(left.CBegin(), left.CEnd(), right.
            CBegin(), right.CEnd(), C());
    }
}
```