#ifndef \_TREESET\_H\_

#define \_TREESET\_H\_

#include <iostream>

#include <memory>

#include <sstream>

#include "BinaryNode.h"

#include "Set.h"

using namespace std;

template <class Thing>

class TreeSet : public Set <Thing>

{

public:

void add(Thing entry);

void remove(Thing entry);

bool contains(Thing object);

int getlength(Thing object);

friend ostream& operator<<(ostream&, Thing&);

void inprint(ostream& out, shared\_ptr<const BinaryNode<Thing>> tree);

private:

shared\_ptr<BinaryNode<Thing>> datum;

static void add(Thing entry, shared\_ptr<BinaryNode<Thing>> datum);

static void remove(Thing entry, shared\_ptr<BinaryNode<Thing>> datum);

static bool contains(Thing object, shared\_ptr<BinaryNode<Thing>> datum);

static int getlength(Thing object, shared\_ptr<BinaryNode<Thing>> datum);

};

template <typename Thing>

void inprint(ostream& out, shared\_ptr<const BinaryNode<Thing>> datum)

{

if(datum == nullptr)

{

}

else

{

inprint(out, datum->left);

out << datum->Character;

inprint(out, datum->right);

}

}

template <typename Thing>

void TreeSet<Thing>::add(Thing entry)

{

if(entry == nullptr)

{

entry = make\_shared<BinaryNode<Thing>>(entry);

}

else

{

if(entry < datum)

{

if(datum->left==nullptr)

{

entry->left = make\_shared<BinaryNode<Thing>>(entry);

}

else

{

add(entry->left, datum);

}

}

if(datum->right==nullptr)

{

entry->right = make\_shared<BinaryNode<Thing>>(entry);

}

else

{

add(entry->right, datum);

}

}

}