Exercise 6. *Deadline: 20 April 2009.*

COURSE OF C++ PROGRAMMING LANGUAGE

BINARY SEARCH TREE

University of Wrocław
Institute of Computer Science *Paweł Rzechonek*

Exercise

In computer science, a *binary search tree* (BST) is a binary tree data structure which has the following properties:

- Each node (item in the tree) has a distinct value.
- Both the left and right subtrees must also be binary search trees.
- The left subtree of a node contains only values less than the node's value.
- The right subtree of a node contains only values greater than the node's value.

The major advantage of binary search trees over other data structures is that the related sorting algorithms and search algorithms such as in–order traversal can be very efficient. Binary search trees are a fundamental data structure used to construct more abstract data structures.

Your task will be to implement the binary search tree in C++. Divide your code with the definition of BST into two parts: a single node of BST (an item in the tree) and a wrapper class to the tree structure (an interface). Hide the definition of node (class `Node`) and show only the definition of wrapper class (`BST`). This is the header file:

```
# ifndef _zad6bst_h_

# define _zad6bst_h_

# include <iostream>
# include <string>

using namespace std;

typedef string TYPE;

class Node;

class BST
{
    Node *tree;
public:
    BST ();
    BST (const BST &bst);
    ~BST ();
public:
    const TYPE * search (const TYPE &x) const;
    bool insert (const TYPE &x);
    bool remove (const TYPE &x);
public:
    int size () const;
    bool empty () const;
    void clear ();
public:
    const TYPE & min () const throw(string);
```

```
        const TYPE & max () const throw(string);
    public:
        void inorder (ostream &out) const;
    };

    # endif
```

Implement all methods and constructors of the class `BST`. Write the definition of class `Node` and its methods (most of them should be recursive) in the source file.

The methods `BST::min()` and `BST::max()` should throw an exception (the `string` object) always if the tree is empty.

Finally write a program, which:

- reads data line by line from the standard input;
- writes to the standard output unique data, in the lexicographical order.

Use the BST for maintaining the lines in your program.

Suggestion

Partition your code into the header and source files.

Hint

Consider a simple program that reads data line by line from the standard input. It can be helpful to complete your task.

```
#include <iostream>
#include <string>

using namespace std;

int main(int argc, char *argv[])
{
    string ln;
    for (getline(cin,ln); cin; getline(cin,ln))
        cout << "[" << ln << "]" << endl;
}
```

Hint

Some information about the binary search tree can be found on the webpage:

`http://en.wikipedia.org/wiki/Binary_search_tree`