Project 4

For this project you are to submit your well documented WordData class from the second half of Lab 7 and your well documented OrderedSet template class from Lab 14.

Specifications

1. The class descriptions for WordData and Ordered Set are attached. Sample function documentation is also attached.

2. The OrderedSet class Insert function should place the new value in the set to maintain the data entries in ascending order. You should only use the < operator to determine the order of the data entries. If the value cannot be inserted in the set due to lack of memory space or because the value is already in the set, this function should return false; otherwise, it should return true. You may use InsertFirst and/or InsertLast if you wish.

3. Delete should remove the specified data entry from the set. If the value is not in the set this function should return false; otherwise, it should return true. You may use DeleteFirst and/or DeleteLast if you wish.

4. Clear should remove all items from the set. (Use a delete function.)

5. isEmpty should return true if the set is empty and false if there are entries in the set.

6. isIn should return true if the specified data value is in the set and false if the value is not in the set.

7. The operator + should return a new set containing the union of the set pointed to by this and the set passed as other. Your implementation should take advantage of other OrderedSet functions.

8. The operator * should return a new set containing the intersection of the set pointed to by this and the set passed as other. Your implementation should take advantage of other OrderedSet functions.

9. All other functions and operators should be implemented as discussed in class.

10. Appropriate exception handling should be included. All exceptions should be thrown using the OrderedSet Exception class. The application program should handle exceptions when they are thrown; this means that your functions do not need to “try” and “catch”, only “throw”. The indexing ([]) operator should include the problematic subscript value in the exception messages. All other exception messages should be clear and accurate.
11. Please make sure you have tested all parts of your OrderedSet class. You should test using (as a minimum) sets of integers, strings, and a user defined class such as fraction class we used in Labs 9 and 10.

12. Make sure you have tested all parts of your OrderedSet class using your WordData class.

**Date Due:** Sunday 15 April 2016; 11:59 pm.

**To Turn In:** A file called yourlastnameP4.tgz containing your tarred and zipped yourlastnameP4 folder. The folder should contain: OrderedSet.hpp, WordData.h, and WordData.cpp and README.txt.

```c++
#ifndef WORDDATA
#define WORDDATA

#include <cstring>
#include <string>
#include <iostream>
#include <iomanip>
using namespace std;

class WordData {
public:
    WordData ();
    WordData (const WordData & OtherWord);
    ~WordData ();
    string GetWord () const;
    int GetNumVowels () const;
    int GetNumConsonants () const;
    int GetNumDigits () const;
    int GetNumSpecialChars () const;
    WordData & operator = (const WordData & OtherWord);
    bool operator == (const WordData & OtherWord) const;
    bool operator < (const WordData & OtherWord) const;
    WordData operator + (const WordData & OtherWord) const;
    friend istream & operator >> (istream & ins, WordData & w);
    friend ostream & operator << (ostream & outs, const WordData & w);
    void SetWord (string InWord);
    void WriteData (ostream & outs) const;

private:
    char * word;
    int vowels;
    int consonants;
    int digits;
    int specials;
};
#endif
```
/***************************************************************************/
* Project: Project 4
* File: OrderedSet.hpp
* Author:
* Project:
* Description:
*******************************************************************************/

#ifndef ORDEREDSET_HPP
#define ORDEREDSET_HPP

#include <iostream>
#include <sstream>
#include <cstdlib>
using namespace std;

dtype {FORWARD, BACKWARD};

template <class ST> class OrderedSet;
template <class ST> ostream & operator <<
   (ostream & outs, const OrderedSet<ST> & S);

template <class ST>
class OrderedSet
{
   private:
      class Node
      {
         public:
            Node ();
            ST data;
            Node * next;
            Node * prev;
      };

   public:
      class Iterator
      {
         public:
            Iterator ();
            Iterator (Node * NP);
            const ST operator * () const;
            Iterator operator ++ ();
            Iterator operator ++ (int);
            Iterator operator -- ();
            Iterator operator -- (int);
            bool operator == (const Iterator & other) const;
            bool operator != (const Iterator & other) const;

            private:
               Node * current;
      };

      class Exception
      {
         public:
            Exception ();
            Exception (string msg);
            Exception (int idx);
      };

#endif // ORDEREDSET_HPP
string Message () const;
private:
    string message;
};

OrderedSet ();
OrderedSet (const OrderedSet & other);
~OrderedSet ();
OrderedSet & operator = (const OrderedSet & other);
bool operator == (const OrderedSet & other);
int Size () const;
void Clear ();
bool IsEmpty () const;
bool IsIn (const ST & value) const;
OrderedSet operator + (const OrderedSet & other) const;
OrderedSet operator * (const OrderedSet & other) const;
friend ostream & operator << <> (ostream & outs, const OrderedSet<ST> & S);
bool Insert (const ST & value);
bool Delete (const ST & value);
void Forward (void function (const ST & param));
void Backward (void function (const ST & param));
void SetDirection (dtype D);
dtype GetDirection () const;
Iterator begin () const;
Iterator rbegin () const;
Iterator end () const;
Iterator rend () const;
ST operator [] (const int & index) const;
private:
    bool InsertFirst (const ST & value);
    bool InsertLast (const ST & value);
    bool DeleteFirst ();
    bool DeleteLast ();
    int direction;
    Node * first;
    Node * last;
    int size;
};

/**********************************************************
* Function: Node()                                         *
* Parameters: none                                          *
* Return value: none                                        *
* Description: Default constructor for the Node class. This function will *
* initialize the next and prev pointers of the Node class to NULL. The data *
* value initialization is left to the data type class. Data values of *
* primitive types may, or may not, be initialized (compiler dependent). *
**********************************************************/

template <class ST>
OrderedSet<ST>::Node::Node ()
{
    next = NULL;
    prev = NULL;
}

... More documentation and functions here ...

#endif