```
#include "SLNode.h"

#include <cstdlib>
#include <string>
#include <sstream>

using namespace std;

#ifndef SLIST_H
#define SLIST_H

template<typename T>
class SList
{
    public:

    /***** constructor/destructor definitions *****/

        SList()
        : head( NULL ), size( 0 )
        {
            /* empty */
        }

        ~SList()
        {
            this->clear();
        }

    /***** public function definitions *****/

        void insertHead( T headValue )
        {
            if( size > 0 )
            {
                SLNode<T> *selectedNode = this->head;

                this->head = new SLNode<T>( headValue );
                size++;
                head->setNextNode( selectedNode );
            }
            else
            {
                this->head = new SLNode<T>( headValue );
                size++;
            }
        }

        void insertTail( T tailValue )
        {
```

```
        if( size > 0 )
        {
            SLNode<T>* selectedNode = head;

            while( selectedNode->getNextNode() != NULL )
            {
                selectedNode = selectedNode->getNextNode();
            }

            selectedNode->setNextNode( new SLNode<T>( tailValue ) );
        }
        else
        {
            head = new SLNode<T>( tailValue );
        }

        size++;
    }

    void removeHead()
    {
        if( size > 0 )
        {
            SLNode<T>* selectedNode = (this->head)->getNextNode();

            delete this->head;
            this->head = selectedNode;
            size--;
        }
    }

    void removeTail()
    {
        if( size > 1 )
        {
            SLNode<T>* selectedNode = head;

            while( (selectedNode->getNextNode())->getNextNode() != NULL )
            {
                selectedNode = selectedNode->getNextNode();
            }

            delete selectedNode->getNextNode();
            selectedNode->setNextNode( NULL );
            size--;
        }
        else if( size == 1 )
        {
            delete this->head;
            size--;
```

```
            }
        }

        void insert( T insertValue )
        {
            if( size > 0 )
            {
                SLNode<T>* selectedNode     = head;
                SLNode<T>* insertionNode    = NULL;

                while( selectedNode->getNextNode() != NULL )
                {
                    if( ( ( selectedNode->getContents() < insertValue ) && ( selectedNode->getNextNode()->getContents() >
                    insertValue ) )
                            || ( selectedNode->getContents() == insertValue ) )
                    {
                        insertionNode   = new SLNode<T>( insertValue );
                        insertionNode->setNextNode( selectedNode->getNextNode() );
                        selectedNode->setNextNode( insertionNode );
                        size++;

                        return;
                    }
                    else
                    {
                        selectedNode = selectedNode->getNextNode();
                    }
                }

                if( selectedNode->getContents() <= insertValue )
                {
                    insertTail( insertValue );
                    return;
                }
                else if( selectedNode->getContents() > insertValue )
                {
                    insertHead( insertValue );
                    return;
                }
            }
            else
            {
                head = new SLNode<T>( insertValue );
                size++;
            }
        }

        bool removeFirst( T removeValue )
        {
            if( size != 0 )
```

```
150             {
151                 if( head->getContents() == removeValue )
152                 {
153                     removeHead();
154                     return true;
155                 }
156             }
157             else
158             {
159                 return false;
160             }
161 
162             if( size >= 2 )
163             {
164                 SLNode<T>* selectedNode     = head;
165                 SLNode<T>* removeNode       = NULL;
166 
167                 if( !( selectedNode->getNextNode()->getContents() == removeValue ) )
168                 {
169                     while( selectedNode->getNextNode()->getNextNode() != NULL )
170                     {
171                         if( selectedNode->getNextNode()->getContents() == removeValue )
172                         {
173                             break;
174                         }
175 
176                         selectedNode = selectedNode->getNextNode();
177                     }
178                 }
179 
180                 if( selectedNode->getNextNode()->getContents() == removeValue )
181                 {
182                     if( selectedNode->getNextNode()->getNextNode() == NULL )
183                     {
184                         removeTail();
185                         return true;
186                     }
187                     else
188                     {
189                         removeNode = selectedNode->getNextNode();
190                         selectedNode->setNextNode( removeNode->getNextNode() );
191 
192                         delete removeNode;
193                         removeNode = NULL;
194 
195                         size--;
196                         return true;
197                     }
198                 }
199             }
```

```
200
201             return false;
202         }
203
204         bool removeAll( T removeValue )
205         {
206             if( size > 0 )
207             {
208
209                 SLNode<T>* selectedNode     = this->head;
210                 SLNode<T>* removeNode       = NULL;
211
212                 while( selectedNode->getContents() == removeValue )
213                 {
214                     removeNode      = selectedNode;
215                     this->head      = removeNode->getNextNode();
216
217                     delete removeNode;
218                     size--;
219
220                     selectedNode = this->head;
221                     if( selectedNode == NULL )
222                     {
223                         return true;
224                     }
225                 }
226
227                 SLNode<T>* followerNode = NULL;
228                 if( selectedNode != NULL )
229                 {
230                     while( selectedNode->getNextNode() != NULL )
231                     {
232                         followerNode    = selectedNode;
233                         if( selectedNode->getNextNode()->getContents() == removeValue )
234                         {
235                             removeNode  = selectedNode->getNextNode();
236
237                             selectedNode->setNextNode( removeNode->getNextNode() );
238
239                             delete removeNode;
240                             size--;
241
242                             continue;
243                         }
244
245                         selectedNode = selectedNode->getNextNode();
246                     }
247
248                     if( selectedNode->getContents() == removeValue )
249                     {
```

```
                        removeNode = selectedNode;
                        followerNode->setNextNode( NULL );

                        delete removeNode;
                        size--;
                    }
                }

                if( removeNode != NULL )
                {
                    return true;
                }
            }

            return false;
        }

        void clear()
        {
            if( size > 1 )
            {
                SLNode<T>* selectedNode;

                do {
                    selectedNode = (this->head)->getNextNode();
                    delete this->head;
                    this->head = selectedNode;

                } while( (this->head)->getNextNode() != NULL );
            }

            if( size >= 1 )
            {
                delete this->head;
                size = 0;
            }

        }

        string toString() const
        {

            if( size > 0 )
            {
                stringstream ss;

                if( (this->head)->getNextNode() == NULL )
                {
                    ss << (this->head)->getContents();
                    return ss.str();
```

```
                }

                SLNode<T>* selectedNode = head;

                do {
                    ss << selectedNode->getContents() << ",";
                    selectedNode = selectedNode->getNextNode();

                } while( selectedNode->getNextNode() != NULL );

                ss << selectedNode->getContents();

                return ss.str();
            }
            else
            {
                return "";
            }
        }

    /***** accessor/mutator function definitions *****/
        unsigned int getSize() const
        {
            return this->size;
        }

    private:

    /***** private variable declarations *****/

        SLNode<T>* head;
        unsigned int size;

};

#endif
```