```cpp
/*
 * proj3
 * Working example of a templated doubly linked list of type int.  Loads a specially formatted
 * commands file which, when read by the program, constructs a doubly linked list with a particular
 * configuration.  Commands are reported to the console.
 *
 * Programmer:  Chad Philip Johnson
 * Date Created:  Thursday, March 28th, 2013
 * Last Date Modified:  Sunday, April 14th, 2013
 *
 * Sources Used:
 *      DLNode.h
 *          - to create instances of individual nodes containing values of the supplied type within a doubly linked list
 *      DLList.h
 *          - the doubly linked list apparatus
 */

#pragma once

#include <cstdlib>
#include <string>
#include <iostream>
#include <fstream>

#include "DLNode.h"
#include "DLList.h"

using namespace std;

/**
 * Processes the program commands input from a formatted data file.
 * @param pobjDLListDriver The instance of the doubly linked list to be built or modified.
 * @param charCommand The command to be issued to the doubly linked list.
 * @param strInput The value to be added to the doubly linked list.
 */
template<typename T>
void processInput( DLList<T>*& pobjDLListDriverList, char& charCommand, string strInput = "" );

int main( int argc, char* argv[] )
{
    DLList<int>* pobjDLListDriverList   = NULL;

    if( argc > 1 )
    {
        ifstream objifstreamInputFile( argv[1] );
        if( objifstreamInputFile.good() )
        {
            string strNextLine  = "";

            while( getline( objifstreamInputFile, strNextLine ) )
```

```
            {
                if( strNextLine.length() == 1 )
                {
                    processInput( pobjDLListDriverList, strNextLine[0] );
                }
                else if( strNextLine.length() > 1 )
                {
                    processInput( pobjDLListDriverList, strNextLine[0], strNextLine.substr( 2 ) );
                }
            }

            objifstreamInputFile.close();
        }
        else
        {
            cout << "The file " << argv[1] << " does not exist." << endl;
        }
    }
    else
    {
        cout << "Usage of proj3.exe:" << endl;
        cout << "\t" << "proj3.exe NAME_OF_FILE" << endl;
    }

}

template<class T>
void processInput( DLList<T>*& pobjDLListDriverList, char& charCommand, string strInput )
{

    switch( charCommand )
    {
        case '#':
            break;

        case 'C':
        case 'c':
            if( pobjDLListDriverList != NULL )
            {
                delete pobjDLListDriverList;
                pobjDLListDriverList    = NULL;
            }
            pobjDLListDriverList = new DLList<int>;

            cout << "LIST CREATED" << endl;
            break;
    }

    if( pobjDLListDriverList != NULL )
    {
```

```
        switch( charCommand )
        {
            case 'X':
            case 'x':
                pobjDLListDriverList->clear();
                cout << "LIST CLEARED" << endl;
                break;

            case 'D':
            case 'd':
                delete pobjDLListDriverList;
                pobjDLListDriverList    = NULL;
                cout << "LIST DELETED" << endl;
                break;

            case 'I':
            case 'i':
                pobjDLListDriverList->insert( atoi( strInput.c_str() ) );
                cout << "VALUE " << strInput << " INSERTED" << endl;
                break;

            case 'F':
            case 'f':
                pobjDLListDriverList->pushFront( atoi( strInput.c_str() ) );
                cout << "VALUE " << strInput << " ADDED TO HEAD" << endl;
                break;

            case 'B':
            case 'b':
                pobjDLListDriverList->pushBack( atoi( strInput.c_str() ) );
                cout << "VALUE " << strInput << " ADDED TO TAIL" << endl;
                break;

            case 'A':
            case 'a':
                try
                {
                    cout << "VALUE " << pobjDLListDriverList->getFront() << " AT HEAD" << endl;
                }
                catch( const char* e )
                {
                    cout << e << endl;
                }
                break;

            case 'Z':
            case 'z':
                try
                {
                    cout << "VALUE " << pobjDLListDriverList->getBack() << " AT TAIL" << endl;
```

```
                }
                catch( const char* e )
                {
                    cout << e << endl;
                }
                break;

            case 'T':
            case 't':
                try
                {
                    if( pobjDLListDriverList->getSize() == 0 )
                    {
                        throw "LIST EMPTY";
                    }
                    pobjDLListDriverList->popFront();
                    cout << "REMOVED HEAD" << endl;
                }
                catch( const char* e )
                {
                    cout << e << endl;
                }
                break;

            case 'K':
            case 'k':
                try
                {
                    if( pobjDLListDriverList->getSize() == 0 )
                    {
                        throw "LIST EMPTY";
                    }
                    pobjDLListDriverList->popBack();
                    cout << "REMOVED TAIL" << endl;
                }
                catch( const char* e )
                {
                    cout << e << endl;
                }
                break;

            case 'E':
            case 'e':
                if( pobjDLListDriverList->removeAll( atoi( strInput.c_str() ) ) )
                {
                    cout << "VALUE " << strInput << " ELIMINATED" << endl;
                }
                else
                {
                    cout << "VALUE " << strInput << " NOT FOUND" << endl;
```

```
                }
                break;

            case 'R':
            case 'r':
                if( pobjDLListDriverList->removeFirst( atoi( strInput.c_str() ) ) )
                {
                    cout << "VALUE " << strInput << " REMOVED" << endl;
                }
                else
                {
                    cout << "VALUE " << strInput << " NOT FOUND" << endl;
                }
                break;

            case 'G':
            case 'g':
                if( pobjDLListDriverList->get( atoi( strInput.c_str() ) ) )
                {
                    cout << "VALUE " << strInput << " FOUND" << endl;
                }
                else
                {
                    cout << "VALUE " << strInput << " NOT FOUND" << endl;
                }
                break;

            case 'N':
            case 'n':
                cout << "LIST SIZE IS " << pobjDLListDriverList->getSize() << endl;
                break;

            case 'P':
            case 'p':
                try
                {
                    if( pobjDLListDriverList->getSize() == 0 )
                    {
                        throw "LIST EMPTY";
                    }
                    cout << (*pobjDLListDriverList);
                }
                catch( const char* e )
                {
                    cout << e << endl;
                }
                break;
        }
    }
    else if( charCommand != '#' )
```

```
    {
        cout << "MUST CREATE LIST INSTANCE" << endl;
    }
}

```