Course: Programming Fundamentals – **ENCM 339**
Lab #: Lab 9
Instructor: S. Norman
Student Name: **Mitchell Sawatzky**
Lab Section: **B02**
Date Submitted: **Dec 1, 2015**

Exercise C

```
void Lab9List::remove_1st(const ListItem& theItem) {
    if (headM == 0)
        return;
    Node* p = headM;
    Node* prev = headM;
    while (p->itemM != theItem && p->nextM) {
        prev = p;
        p = p->nextM;
    }
    if (p->itemM == theItem) {
        (p == headM ? headM : prev->nextM) = p->nextM;
        delete p;
    }
}

void Lab9List::copy(const Lab9List& source) {
    destroy();
    if (source.headM) {
        Node* index = source.headM;
        headM = new Node;
        Node* p = headM;
        do {
            p->itemM = index->itemM;
            p->nextM = (index->nextM ? new Node : 0);
            p = p->nextM;
            index = index->nextM;
        } while (index);
    }
}

void Lab9List::destroy() {
    if (headM) {
        Node* index = headM;
        while (index->nextM) {
            Node* ind = index->nextM;
            delete index;
            index = ind;
        }
        headM = 0;
    }
}
```

```
Mitchell@ttys000 01:09 {0} [lab9]$ ./test.out
list1 - created:
list is empty...

list1 - after a call to push_front:
50

list2 - created:
300  450

list2 - after several calls to push_front has:
300  220  330  440  550  300  450

------------------ Function test_removing started ------------

list: expected to have seven nodes:  300  220  330  440  550  300  450
actual output:
  300  220  330  440  550  300  450

After two removes - expected to display:  300  330  440  300  450
actual output:
  300  330  440  300  450

After one more remove - expected to display:  330  440  300  450
actual output:
  330  440  300  450

After another remove - expected to display:  330  440  450
actual output:
  330  440  450

Last remove - still expected to display:  330  440  450
actual output:
  330  440  450

------------------ Function test_copying started --------------

After removing several nodes in test_removing, list must have: 330  440  450

list1 - expected to display: 330  440  450
actual output:
  330  440  450

list - after removing 330 - expected to display:  440  450
actual output:
  440  450

list1 - still expected to dispay:  330  440  450
actual output:
  330  440  450

list2 - expected to display: 330  440  450
actual output:
  330  440  450
```

```
list1 - expected to display:  989  330  440  450
actual output:
  989  330  440  450

list2 - still expected to display:  330  440  450
actual output:
  330  440  450

list3 - expected to display:  1000  2000  1234
actual output:
  1000  2000  1234

list4 - expected to be empty.
actual output:
list is empty...

list3 - is now expected to be empty.
actual output:
list is empty...
```

Exercise D

```
// lab9_ExD.cpp
// ENCM 339 - FALL 2015 - LAB 9 - EXERCISE D

#include <iostream>
using namespace std;

void insertion_sort(int *int_array, int n, int sort_order);
/* REQUIRES
 *    n > 0.
 *    1 <= sort_order && sort_order <= 2
 *    Array elements int_array[0] ... int_array[n - 1] exist.
 * PROMISES
 *    If sort_order == 1 values of array are rearranged in
ascending order.
 *    If sort_order == 2 values of array are rearranged in
descending order.
 */


int main(int argc, char** argv)
{
    int sort_order;
    if (argc > 2) {
        cerr << "Usage: Too many arguments on the command line"
<< endl;
```

```
        exit(1);
    } else {
        if (argc == 1)
            sort_order = 1;
        else {
            if (strcmp(argv[1], "-a") == 0 || strcmp(argv[1], "-
A") == 0) {
                sort_order = 1;
            } else if (strcmp(argv[1], "-d") == 0 ||
strcmp(argv[1], "-D") == 0) {
                sort_order = 2;
            } else {
                cerr << "Usage: Invalid entry for the command
line option." << endl;
            }
        }

    }

    int a[] = { 413, 282, 660, 171, 308, 537 };

    int n_elements = sizeof(a) / sizeof(int);

    cout << "Here is your array of integers before sorting: \n";
    for(int i = 0; i < n_elements; i++)
        cout <<  a[i] << endl;
    cout << endl;

    insertion_sort(a, n_elements, sort_order);

    if(sort_order == 1)
        cout << "Here is your array of integers after ascending
sort:  \n" ;
    else if(sort_order == 2)
        cout << "Here is your array of integers after descending
sorting:  \n" ;

    for(int i = 0; i < n_elements; i++)
        cout << a[i] << endl;

    return 0;
}

void insertion_sort(int *a, int n, int sort_order)
{
    int i;
    int j;
    int value_to_insert;

    if(sort_order == 1) {

        for (i = 1; i < n; i++) {
            value_to_insert = a[i];

            /* Shift values greater than value_to_insert. */
            j = i;
            while ( j > 0 && a[j - 1] > value_to_insert  ) {
                a[j] = a[j - 1];
                j--;
            }

            a[j] = value_to_insert;
        }

    }
    else {

        for (i = 1; i < n; i++) {
            value_to_insert = a[i];

            /* Shift values less than value_to_insert. */
            j = i;
            while ( j > 0 && a[j - 1] < value_to_insert  ) {
                a[j] = a[j - 1];
                j--;
            }

            a[j] = value_to_insert;
        }
    }
}
```

```
Mitchell@ttys000 01:38 {0} [lab9]$ ./sort -a
Here is your array of integers before sorting:
413
282
660
171
308
537
```

```

Here is your array of integers after ascending sort:
171
282
308
413
537
660
Mitchell@ttys000 01:39 {0} [lab9]$ ./sort -d
Here is your array of integers before sorting:
413
282
660
171
308
537

Here is your array of integers after descending sorting:
660
537
413
308
282
171
```

Exercise E

```

void append_strings (const char** string_array, int n, char**
appended_string) {
     int j, i;
     for (i = j = 0; i < n; i++) {
        int k = 0;
        while (string_array[i][k] != '\0')
            k++;
        j += k;
     }
     char s[j+1];
     for (j = i = 0; i < n; i++) {
        int k = 0;
        while (string_array[i][k] != '\0') {
            s[j] = string_array[i][k];
            k++;
            j++;
        }
     }
     s[j] = '\0';
     *appended_string = s;
}
```

```
Mitchell@ttys000 02:10 {0} [lab9]$ ./test.out
The 6 strings are:
Red.
pink.
almond.
white.
Law.
cup

Expected to display: Red.pink.almond.white.Law.cup
Red.pink.almond.white.Law.cup
```