University of Wollongong

School of Computer Science and Software Engineering

CSCI124 **Applied Programming** **Spring 2012**

Assignment 4 **(Due: Week 10, Wednesday 3 October)** **6 marks**

Aim:

This assignment is to familiarise you with the use of container classes in your programs.

On completion you should know how to:

- Write container class objects and overloaded operators.
- Implement programs incrementally to minimise debugging.
- Custom design container class objects for software applications.

Requirements:

Complete the following class objects by writing the code in the named files. Test that your files compile and run on UNIX prior to submission.

Part A - 4 marks

a) Implement all public member functions of the linked list class declared below in files List.h and List.cpp. Note: RemoveHead() and RemoveTail() return true if an item is successfully removed from the list or false if unsuccessful (ie the list is empty). Also, implement the main() function given below in main1.cpp. Modify the main() function by inserting code for testing all the list's member functions (like we did in Ass3 for class CardSet). Your main() should ensure all test cases are executed. (eg add item to head of empty list, add item to tail of non-empty list, etc). Note: Make sure you put pre-processor directives in List.h to prevent it being compiled more than once.

```
struct Node
{
    int Item;
    Node *Next;
};

class LinkedList
{
    public:
        LinkedList();
        ~LinkedList();
        void AddHead(int Item);     // adds item to head
        bool RemoveHead(int &Item);// removes item from head
        void AddTail(int Item);     // adds item to tail
        bool RemoveTail(int &Item);// removes item from tail
        void Print();               // prints list. eg 12 34 21 26
    private:
        Node *Head;
};

int main()
{
    LinkedList L;

    // Put your code for testing your list here

    return 0;
}
```

Part A continued…

b) In your linked list from Question 1, also implement a copy constructor that makes a deep copy of the list argument.

c) Also implement an assignment operator. Your assignment operator should ensure multiple assignments are possible. eg the following statements should make A and B identical to C. Ensure there are no memory leaks and all lists have their own separate memory. You should not assume the left-hand operand is always empty.

```
List A, B, C;
for(int i=0; i<10; i++)
C.AddHead(i);
A = B = C;
```

d) Modify your main() so that the copy constructor and assignment operator are adequately tested.

Part B – 2 marks

The following program will generate a negative square root error when the number entered is less than 0. Rewrite this program so that it will throw an exception from within DoSquareRoot() when this situation occurs. This exception should be tried and caught in main() as explained in lectures. Save your program in main2.cpp.

```
#include <iostream>
#include <cmath>
using namespace std;

double DoSquareRoot(double);

int main()
{
     double Number, Result;

     // Request a numbers from the user
     cout << "Please enter a number:";
     cin >> Number;

     Result = DoSquareRoot(Number);

     cout<<"\nThe square root of "<<Number<< " is "<<Result<<"\n";
     return 0;
}

double DoSquareRoot(double Number)
{
     double Result = sqrt(Number);
     return Result;
}
```

Submit:

Submit **all your files** for this assignment by using the submit facility on UNIX ie:

$ submit -u login -c CSCI124 -a 4 List.h List.cpp main1.cpp main2.cpp
where 'login' is your UNIX login ID.

We will compile run, print and mark your assignment. An extension of time for the assignment submission may be granted in certain circumstances. Any request for an extension of the submission deadline can be made to the Subject Coordinator or via academic consideration on SOLS before the submission deadline. Supporting documentation should accompany the request for an extension. Late assignment submissions without granted extension will be marked but the points awarded will be reduced by 1 mark for each day late.