

University of Washington Bothell

CSS 342: Data Structures, Algorithms, and Discrete Mathematics

Program 3: Linked Lists

Purpose

This programming assignment will require the usage of dynamic memory, pointers and linked lists. It will also require the usage of File IO and templates. Also, required is a good understanding of overloading assignment and copy constructors. Learnings from previous assignments, especially operator overloading will also be required.

Problem: The List.

You will build a class for a fully ordered list. The list class, called List342, will be templated so that different types of objects can be stored in it.

```
List342<string> aListOfStrings;  
List342<int> aListOfInts;  
List342<Child> children;  
List342<MyRandomObj> myRO;
```

The following member functions should be created for the List342 class. **Please make sure the signatures are exact or points will be deducted as the test programs will not compile.** Also make sure that the proper constructors/destructors are created for the class.

bool BuildList(string fileName) : Given a string which represents a file, open the file, read the objects from file, and build an ordered list. Note that BuildList puts the responsibility on the Object class to know how to read from a file. That is, do not have object specific logic in the implementation. You can insert each item into the list as you read it from the file. If a list already has items, add the new items to the list.

Assume that the list passed in is well-formed for the object being read into.

bool Insert(ItemType *obj) : Insert an object in the list in the correct place. Return true or false depending on whether the insert was successful. Duplicates of an object are not allowed. If there is a duplicate the function should return false. Note that a pointer to the object is passed in but the Insert should create a copy of the object to keep in the list.

bool Remove(ItemType target, ItemType &result): Remove the target item from the list. Return the item using a second parameter that is passed in by reference. If the item was found and removed return true, else return false.

bool Peek(ItemType target, ItemType &result) const: The same as Remove except the item is not removed from the list. Again, the second item is returned by reference.

bool isEmpty() const: Is the list empty?

void DeleteList(): Remove all items from the list. Deallocate all memory appropriately. This includes not only the nodes in the list but the items being pointed to by the nodes.

bool Merge(List342 &list1): Takes a sorted list and merges into the calling sorted list (no new memory should be allocated). At termination of function, the list passed in (list1) should be empty (unless it is the calling list). No duplicates are allowed.

The following operators should be overloaded for the List342<>. Please make sure signatures are correct on overloads.

+, += : This should add two lists together. The lists are assumed to be sorted and the returned list must also be sorted. Use an efficient sort algorithm and avoid unnecessary data allocations. Duplicates are not allowed and expected.

<< : Display the list and only the List object, no extra blanks, no tabs, no endl. Display the items on the list in increasing order. See examples below.

== and != : check for equality or inequality.

= : Assignment. Make a deep copy of all new memory.

Details on testing:

We will utilize multiple different types to test the list. For instance, we will create a list of ints and a list of strings. As List342<> is templated we will also test with a set of objects we create. For instance, I will test your list with a Bird class—I will make sure I have the appropriate operators overloaded in that class so that it will work with your list.

In order for you to test your List342<> I have some code below which utilizes a Child class.

The child class contains a first name (string), last name (string), and age (int). Input is of the form “firstname lastname age” in the file. Assume the test in the file is well formed w/o errors.

Here is an example file:

<i>George Washington</i>	<i>12</i>
<i>Lyndon Johnson</i>	<i>12</i>
<i>Joseph Smith</i>	<i>4</i>
<i>Thomas Paine</i>	<i>17</i>
<i>Thomas Paine</i>	<i>17</i>

The Child class compares names alphabetically by last name first; first name second; and then age. A child is considered a duplicate if all three fields (firstName, lastName, id) are equivalent.

I provide code below which utilizes the Child class and the List342<>.

Other Rules and Hints

- 1) Implement the List342<> class using Nodes defined as below. It can be internal or external to the List342<> class. Keeping the struct definition external to the List342 class may be the easiest solution. However, templating the struct may encounter issues in this way. Therefore, keeping Node in the private area of List342<> is probably best.

Note that the struct contains a pointer to the next Node and a ***pointer*** to the data. This is required for this program.

```
struct Node {
    ItemType *data;
    Node *next;
};
```

- 2) Due to vicissitudes of different IDEs and templating we will be factoring the code into files a little sub-optimally. Put all code into the List342.h file—this includes the implementation (.cpp) as well as the .h.

TURN IN (in a .zip file):

- list342.h and list342.cpp (for the .cpp all code could be commented out as it will be in the list342.h file)
- An implementation of Child class (child.cpp/child.h) as described above
- Output of YOUR program running the Example code below
- The .exe if built/tested on windows OR the a.out if built/tested on Linux

EXAMPLE OUTPUT BELOW:

```
int main()
{
    Child c1("Angie", "Ham", 7), c2("Pradnya", "Dhala", 8),
    c3("Bill", "Vollmann", 13), c4("Cesar", "Ruiz", 6);
    Child c5("Piqi", "Tangi", 7), c6("Pete", "Rose", 13),
    c7("Hank", "Aaron", 3), c8("Madison", "Fife", 7);
    Child c9("Miles", "Davis", 65), c10("John", "Zorn", 4), c11;
    List342<Child> class1, class2, soccer, chess;
    int a = 1, b = -1, c = 13;

    class1.Insert(&c1);
    class1.Insert(&c2);
    class1.Insert(&c3);
    class1.Insert(&c4);
    class1.Insert(&c5);
    class1.Insert(&c6);
    class1.Insert(&c5);
    cout << "class1: " << class1 << endl;

    if (class1.Insert(&c1))
    {
        cout << "ERROR:: Duplicate" << endl;
    }

    class2.Insert(&c4);
    class2.Insert(&c5);
    class2.Insert(&c6);
    class2.Insert(&c7);
    class2.Insert(&c10);
    cout << "Class2: " << class2 << endl;

    class1.Merge(class2);
    class2.Merge(class1);
    class1.Merge(class2);
    class1.Merge(class1);
    cout << "class1 and 2 Merged: " << class1 << endl;

    if (!class2.isEmpty())
    {
        cout << "ERROR:: Class2 should be empty empty" << endl;
    }

    class1.Remove(c4, c11);
    class1.Remove(c5, c11);
    class1.Remove(c11, c11);
    if (class1.Remove(c1, c11))
    {
        cout << "Removed from class1, student " << c11 << endl;
    }
}
```

```

    }
    cout << "class1: " << class1 << endl;

    soccer.Insert(&c6);
    soccer.Insert(&c4);
    soccer.Insert(&c9);
    cout << "soccer: " << soccer << endl;
    soccer += class1;
    cout << "soccer += class1 : " << soccer << endl;

    List342<Child> football = soccer;
    if (football == soccer)
    {
        cout << "football: " << football << endl;
    }
    if (football.Peek(c6, c11))
    {
        cout << c11 << " is on the football team" << endl;
    }
    soccer.DeleteList();
    if (!soccer.isEmpty())
    {
        cout << "ERROR: soccer should be empty" << endl;
    }

    List342<int> numbers;
    numbers.Insert(&a);
    numbers.Insert(&b);
    numbers.Insert(&c);
    cout << "These are the numbers: " << numbers << endl;
    numbers.DeleteList();
    return 0;
}

```

OUTPUT:

```

class1: PradnyaDhala8AngieHam7PeteRose13CesarRuiz6PiqiTangi7BillVollmann13
Class2: HankAaron3PeteRose13CesarRuiz6PiqiTangi7JohnZorn4
class1 and 2 Merged: HankAaron3PradnyaDhala8AngieHam7PeteRose13CesarRuiz6PiqiTangi7BillVollmann13JohnZorn4
Removed from class1, student AngieHam7
class1: HankAaron3PradnyaDhala8PeteRose13BillVollmann13JohnZorn4
soccer: MilesDavis65PeteRose13CesarRuiz6
soccer += class1 : HankAaron3MilesDavis65PradnyaDhala8PeteRose13CesarRuiz6BillVollmann13JohnZorn4
football: HankAaron3MilesDavis65PradnyaDhala8PeteRose13CesarRuiz6BillVollmann13JohnZorn4
PeteRose13 is on the football team

```

These are the numbers: -1113