**My doubly linked list:** My doubly linked list is not circular. It contains a two pointers, one to the previous node and one to the next node. It contains two public members, a value and a count for that value. My list nodes are in the order in which unique values are inserted into the linked list. If an item that is already existing is added to the list, the existing item in the list has its count iterated, otherwise if that item is unique, it is added to the end of the list.

**Pseudocode:**

**erase**

if list is empty, return nothing

have a pointer first point to head

use ptr-> next to traverse the linked list

if at any point m\_val == value

subtract one from count if count is >1

call eraseAll to get rid of that Node if count is 1

if we traversed through the entire linked list, return 0, nothing to erase

**eraseAll**

case 1:its empty – return 0

case 2: theres only one item – delete it and set head / tail to nullptr

traverse the linked list with ptr->next

case 3: delete the head – manage where the head goes update the prev pointer of the next node

case 4: it’s somewhere in the middle of the list – use temp variables to manage the next / prev pointers, successfully delete the node

case 5: its at the end of linked list – delete it and update tail

**destructor**

if empty, return

while (head != tail)

set temp pointer to point to tail, iterate tail up, delete the temp pointer

set head / tail to nullptr

**copy constructor**

if empty, return

create a head, have it point to a new node that is has its values copied from src

use a while loop and use the insert function to add all of the items from src into \*this

**assignment operator**

check for aliasing

copy/paste destructor and copy constructor

**combine**

to insert any one list to another list, use a nested for loop

iterate through the unique size of one list

create a temp ItemType variable and pass it into the get function

insert the temp into the result

case 1: they are all aliases for one another – double the count for everything in result

case 2:result is alias of ms1, or result is not an alias of anything

either way, I set result = ms1 (my copy constructor handles aliasing) and use my get + insert code to add ms2 into result

case 3: result is an alias of ms2

now I add in ms1 instead of ms2 for the get + insert code

**subtract**

the subtract code itself is very similar to the one from combine, but instead of insert, I use erase

case 1: ms1 is alias for ms2, or ms1, ms2, result are all alias for each other

I simply erase everything from result and it will be an empty list

Case 2: ms2 is alias for result

Use copy constructor to store ms2 into a temp Multiset.

Use assignment operator to make result = ms1

Delete ms2 from ms1 as shown below

Case 3: ms1 is aliased with result, or result is not aliased with anything

Either way, I set result = ms1 (copy constructor deals with aliasing) and simply use my get + erase code to delete the values

**Test Code**

I felt the code posted on the website was enough to test insert and size

Testing size, uniquesize, and insert

Multiset x;

x.insert(“haha”);

x.insert(“haha”);

x.insert(“hehehe”);

x.insert(“haha”);

count for haha should be 3, hehehe should be 1, uniquesize should be 2, size should be 4

Testing erase and eraseAll

x.erase(“haha”) returns 1, haha count is 2

x.eraseAll(“haha”) that node is gone

Testing swap

Multiset y;

y.insert(“LOLLLLL”);

y.swap(x);//x now has LOLLLLL and y now has hehehe

Testing copy constructor and assignment operator

Multiset z = y; //should be new linked list with same items as y

x = y; //should be new linked list with same items as y

y = y; //nothing weird should happen

Testing combine

Multiset ms1;

Multiset ms2;

Multiset result;

ms1.insert(“lollipop”);

ms1.insert(“lucky”);

ms1.insert(“lucker”);

ms2.insert(“zzz”);

ms2.insert(“give”);

ms2.insert(“me”);

ms2.insert(“A”);

{insert code for case}

Case1: ms1 and ms2 and result are aliased

Combine(ms1, ms1, ms1); //result should just be ms2’s original value

Case2: result is alias for ms1 or nothing is alias of anything

Combine(ms1, ms2, ms1); //shouldn’t matter

//result should just be combined list of ms1 and ms2

Case3: result is alias for ms2

Combine(ms1, ms2, ms2);

//result should still be combined list of ms1 and ms2

Testing subtract

Multiset ms1;

Multiset ms2;

Multiset result;

ms1.insert(“lollipop”);

ms1.insert(“lucky”);

ms1.insert(“lucky”);

ms1.insert(“lucker”);

ms2.insert(“lollipop”);

ms2.insert(“lucky”);

ms2.insert(“lucker”);

ms2.insert(“AAAA”);

case 1:ms2 is aliased with result and ms1–return empty list

case 2: result is aliased with ms2

case 3: result aliased with ms1 or with nothing – both case 2 and 3 should print ms1 – ms2