Ahsan, Prianna

SID: 704068040

**Project 2 Report: double Trouble = 2^n\*Annoying**

**Description**

Multiset objects hold a double linked list of Node objects. Each Node object has previous and next pointers that link it to adjacent nodes, and holds a value of pre-defined ItemType and a count of the number of values held. The default constructor creates an empty multiset with head and tail pointers set to null, and all size variables set to zero. A populated multiset consists of a linked list of nodes that is neither circular nor preceded by a dummy node.

**Pseudocode for non-member functions**

Subtract

Specification: *If an item appears n1 times in ms1 and n2 times in ms2, then when this function returns, that item must appear exactly n1−n2 times in result if n1 > n2; otherwise, it must not appear in result. You must not assume result is empty when it is passed in to this function; it may not be.*

Psuedocode:

Copy first multiset.

If a value in first multiset exists in second multiset:

Check if count of value in first multiset exceeds the count of value in second multiset:

Subtract second value from first value and store difference in copy.

Set result multiset to copy.

Delete copy to reallocate memory.

Combine

Specification: *When this function returns, result must contain a copy of all the items that appear in ms1 or ms2 or both, and must not contain any other items. (You must not assume result is empty when it is passed in to this function; it might not be.) If an item appears n1 times in ms1 and n2 times in ms2, it must appear exactly n1+n2 times in result.*

Pseudocode:

Copy first multiset.

If any of second multiset’s values are in first multiset:

Find out how many times the value appears in second multiset.

Insert value x times into copy.

Set result multiset to copy.

Delete copy to reallocate memory.

**Test Cases**

Multiset ms; // default constructor

assert(ms.empty()); // check empty function

unsigned long x = 999;

// x unchanged by get failure

assert(ms.get(0, x) == 0 && x == 999);

assert( ! ms.contains(42) ); // check contains function

ms.insert(42); // check insert

ms.insert(42);

// check unique size and size

assert(ms.size() == 2 && ms.uniqueSize() == 1);

assert(ms.get(0, x) == 2 && x == 42); // test get

ms.insert(66);

ms.insert(66);

unsigned long l1;

int n1 = ms.get(1, l1);

// test get

assert((l1 == 66 && n1 == 2) || (l1 == 42 && n1 == 2));

unsigned long l2;

int n2 = ms.get(1, l2);

assert(l2 == l1 && n2 == n1);

assert(ms.count(42) == 2);

ms.erase(42); // test erase

assert(ms.count(42) == 1);

ms.insert(66);

assert(ms.count(66) == 3);

assert(ms.uniqueSize() == 2);

assert(ms.size() == 4);

ms.eraseAll(66);

assert(ms.count(66) == 0);

Multiset ms2;

ms.swap(ms2); // test swap

int p = ms2.count(42);

assert(p == 1);

Multiset ms3(ms2);

int q = ms3.count(42);

assert(q == 1);

ms.insert(86);

ms3 = ms;

assert(ms3.count(86) == 1 && ms.count(66) == 0);

cout << "MS2 dump" << endl;

ms2.dump();

cout << "MS3 dump" << endl;

ms3.dump();

ms.insert(55);

combine(ms3, ms2, ms); // test combine

combine(ms3, ms2, ms3);

assert(ms.count(86) == 1 && ms.count(42) == 1);

cout << "Combined MS dump" << endl;

ms.dump();

subtract(ms, ms2, ms);

assert(ms.count(55) == 0); // test subtract

My test cases for strings consisted of the provided sample code:

Multiset sms;

assert(sms.insert("cumin"));

assert(sms.insert("turmeric"));

assert(sms.insert("cumin"));

assert(sms.insert("coriander"));

assert(sms.insert("cumin"));

assert(sms.insert("turmeric"));

assert(sms.size() == 6 && sms.uniqueSize() == 3);

assert(sms.count("turmeric") == 2);

assert(sms.count("cumin") == 3);

assert(sms.count("coriander") == 1);

assert(sms.count("cardamom") == 0);

All cases passed, and none of the assertions failed. I didn’t test strings further because none of the implemented functions changed, just the typedef.

**Memory Leak Test Results**

**test A** // No dummy nodes.

**test B** // insert(10).

Created 10(0x7fff5fbff910) // temp var created by insert

// temp var passed to find from insert.

Created 10(0x7fff5fbff800) copied from (0x7fff5fbff910)

// find returns NULL, so new node created.

Created 10(0x100100cc0) copied from (0x7fff5fbff800)

// destroy temp vars.

Destroyed 10(0x7fff5fbff800)

Destroyed 10(0x7fff5fbff910)

**test C** // insert(20)

Created 20(0x7fff5fbff908) // temp var created by insert.

// find function.

Created 10(0x7fff5fbff780) copied from (0x100100cc0)

Destroyed 10(0x7fff5fbff780)

Created 20(0x7fff5fbff800) copied from (0x7fff5fbff908)

// find returns NULL, so new node created.

Created 20(0x100100ce0) copied from (0x7fff5fbff800)

// destroy temp vars.

Destroyed 20(0x7fff5fbff800)

Destroyed 20(0x7fff5fbff908)

**test D** // no dummy nodes.

**test E** // insert(30) into an empty multiset.

Created 30(0x7fff5fbff8e8)

Created 30(0x7fff5fbff800) copied from (0x7fff5fbff8e8)

Created 30(0x100100d00) copied from (0x7fff5fbff800)

Destroyed 30(0x7fff5fbff800)

Destroyed 30(0x7fff5fbff8e8)

**test F** // assign ms to ms2.

Created 0(0x7fff5fbff780) (default)

Created 10(0x7fff5fbff700) copied from (0x100100cc0)

Replaced 0(0x7fff5fbff780) by assigning from 10(0x7fff5fbff700) // modifying m\_head

Destroyed 10(0x7fff5fbff700) // deleting temp node

Created 10(0x7fff5fbff710) copied from (0x7fff5fbff780)

Created 10(0x100100d40) copied from (0x7fff5fbff710)

Destroyed 10(0x7fff5fbff710)

Destroyed 10(0x7fff5fbff780)

Created 0(0x7fff5fbff780) (default)

Created 20(0x7fff5fbff700) copied from (0x100100ce0)

Replaced 0(0x7fff5fbff780) by assigning from 20(0x7fff5fbff700)

Destroyed 20(0x7fff5fbff700)

Created 10(0x7fff5fbff690) copied from (0x100100d40)

Destroyed 10(0x7fff5fbff690)

Created 20(0x7fff5fbff710) copied from (0x7fff5fbff780)

Created 20(0x100100d60) copied from (0x7fff5fbff710)

Destroyed 20(0x7fff5fbff710) // deleting temp nodes.

Destroyed 20(0x7fff5fbff780)

Destroyed 30(0x100100d00) // deleting old node.

**test G** // insert(40) into ms2.

Created 40(0x7fff5fbff8e0)

// find function

Created 20(0x7fff5fbff780) copied from (0x100100d60)

Destroyed 20(0x7fff5fbff780)

Created 10(0x7fff5fbff760) copied from (0x100100d40)

Destroyed 10(0x7fff5fbff760)

Created 40(0x7fff5fbff800) copied from (0x7fff5fbff8e0)

// creating new node.

Created 40(0x100100d20) copied from (0x7fff5fbff800)

// deleting temp vars.

Destroyed 40(0x7fff5fbff800)

Destroyed 40(0x7fff5fbff8e0)

**test H** // destructor

Destroyed 40(0x100100d20)

Destroyed 20(0x100100d60)

Destroyed 10(0x100100d40)

Destroyed 20(0x100100ce0)

Destroyed 10(0x100100cc0)

DONE // all nodes removed.