Hanna Co

1. The double linked list is a list of Pair items, when contains a keyType and valueType. It has a head and tail pointer, but isn’t circular, and has no dummy nodes. The list is not in any particular order.
2. Pseudocode
   1. Copy Constructor

initialize all the member variables

make a new pointer (src) that points to head

while src does not point to nullptr

get the values of each Pair in the list

insert it into the new linked list

move on to the next element

* 1. Assignment operator

if they are not the same map

make a duplicate of the source map

swap the values

* 1. erase

if there is only one element, and it is the correct one

create a new pointer to that Pair

set head and tail to nullptr

delete the element

decrement num

if there are more than one, and the target element is the first element

create a new pointer to that Pair

point head to the next element, and its previous to nullptr

delete the element and decrement num

if the target element is the last element

create a new pointer to that Pair

point head to the previous element, and its next to nullptr

delete the element and decrement num

otherwise, traverse the list until you get the element before the target

create a new pointer to the target Pair

set next of target’s previous element to the element after target

set previous of target’s next element to the element before target

delete target and decrement num

* 1. swap

temporary pointer to the source’s head

set source’s head to head

set head to source’s head (temporary pointer)

temporary pointer to the source’s tail

set source’s tail to tail

set tail to source’s tail (temporary pointer)

temp int for source’s num

set source’s num to num

set num to the temp int

* 1. combine

traverse the first map

if the element is not in the second map or the resultant map

add it to the resultant map

if it is in the second map

if their values are the same, add it to the resultant map

otherwise, return false

traverse the second map

if it is not in the resultant map, insert it

* 1. reassign

make result a duplicate of map

traverse the map

get the value of the element, and the value of the element after

assign the value of the current element to the element after

3.

Map m; //default constructor

assert(m.empty()); //make sure it’s empty

ValueType v = -1234.5;

assert(!m.get("abc", v) && v == -1234.5); // v unchanged by get failure

m.insert("xyz", 9876.5);

assert(m.size() == 1); //make sure it was properly inserted

KeyType k = "hello";

assert(m.get(0, k, v) && k == "xyz" && v == 9876.5); //testing that the get function works properly

Map n;

n.insert("hello", 1);

n.insert("world", 2);

ValueType x = 0;

assert(n.size() == 2); //the size function works properly

n.update("hello", 3); //testing the update function works

n.get("hello", x); //puts values at “hello” into x

assert(x == 3); //make sure get function works properly

n.swap(m);

assert(n.size() == 1 && m.size() == 2); //make sure it swapped the sizes

Map a = m; //testing copy constructor

assert(a.size() == m.size()); //the sizes are the same

a = n; //testing assignment operator

assert(a.size() == n.size()); //the sizes are the same

Map one;

one.insert("michelle", 8);

one.insert("adelpha", 7);

one.insert("hanna", 14);

Map two;

two.insert("david", 2);

two.insert("steven", 35);

two.insert("roger", 11);

Map oneMod;

reassign(one, oneMod); //testing reassignment operator

ValueType check1 = 8;

ValueType check2 = 7;

ValueType check3 = 14;

assert(oneMod.get("michelle", check1) && check1 != 8);

assert(oneMod.get("adelpha", check2) && check2 != 7);

assert(oneMod.get("hanna", check3) && check3 != 14);

//testing to make sure that the values aren’t the same as previous

combine(one, two, oneMod); //testing combine

assert(oneMod.size() == one.size() + two.size()); //making sure it combined properly

assert(oneMod.contains("hanna") && oneMod.contains("michelle") && oneMod.contains("adelpha") && oneMod.contains("david") && oneMod.contains("steven") && oneMod.contains("roger"));

//contains all the elements it’s supposed to

Map result;

Map h;

h.insert("sally", 6);

h.insert("bob", 12);

Map j;

j.insert("sally", 6);

j.insert("mary", 7);

combine(h, j, result);

assert(result.size() == 3); //testing that it added sally once

Map test;

Map three;

Map four;

three.insert("this", 1);

three.insert("is", 2);

three.insert("for", 3);

four.insert("rachel", 4);

four.insert("is", 5);

assert(!combine(three, four, test) && test.size() == 1);

//was only able to add one element before returning false