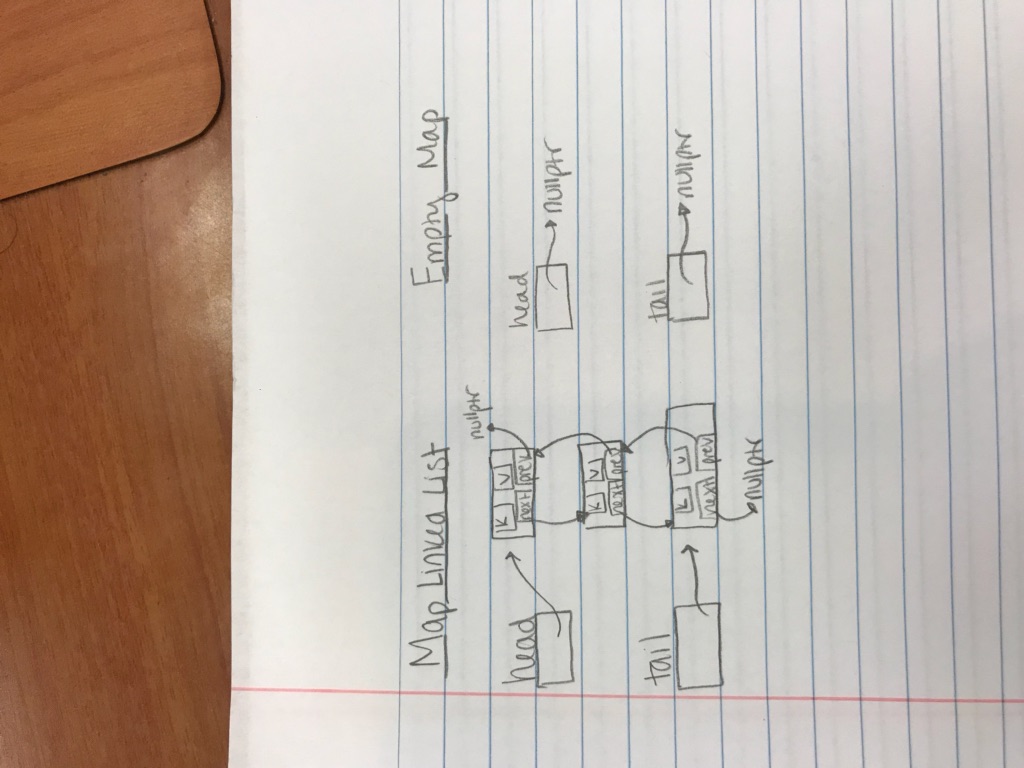
Maeneka Grewal

SID: 204901948

Nachenberg CS32

Project 2 Report

1. The doubly linked list consists of a head pointer node, a tail pointer node, and a node structure that contains the data items (keys and values) as well as next and previous pointers. The linked list of course also had a constructor and destructor. The constructor sets both the head and tail to ‘nullptr’. The destructor traverses through the list and deletes each node until all are deleted. In addition, it contains functions to manipulate the linked list or gather data from it. They are the following: addToFront, addToRear, deleteItem, findItem, changeItem, getItem, traverseTo, and printItems. While the doubly linked list does have a head node and tail node, it is not cyclical and it also does not have a dummy node. The following diagrams show what the linked list looks like.

2. Pseudocode

Overloaded Assignment Operator:

For (traverse through original map)

Traverse to ‘i’ in map, store key and values

Delete item ‘key’ in map

Set original map size to source size

For (traverse through soruce map)

Traverse to ‘i’ in new map, store key and values

Add key and values to rear of original map

Return original map

InsertOrUpate:

If (the map already contains the key)

Send key and new value to ‘changeItem’ function of linked list

Which just updates value to new value

Else

Add new key and value to rear, increment size

Erase:

If (the map contains the key)

Delete the Item from linked list delete

Decrement size and return true

Else

Return false

Combine:

Create temp map variable called ‘combined’ using copy constructor on m1

For (traverse through second map)

Store key and values of second map ‘i’ using get function

If the ‘combined’ map also contains the key

Compare the m2 value and the combined value using get function

If(values do not match)

Boolean matching = false

Erase the key in the combined map

Else

Insert the key and value from m2 that aren’t already in combined

Swap combined and result

Return matching

Subtract:

Create temp map variable called ‘subtracted’ using copy constructor on m1

For (traverse through subtracted)

Store key and values of subtracted using ‘i’ and get function

If m2 contains the key

Erase that key from subtracted

Swap subtracted and result

3. Test Cases

Testing Strings & Doubles

Map m; // test default constructor

assert(m.empty());// test empty function

ValueType v = -1234.5;

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

m.insert("xyz", 9876.5); // testing insert

assert(m.size() == 1); // Testing size

KeyType k = "hello";

assert(m.get(0, k, v) && k == "xyz" && v == 9876.5);

cout << "Passed all tests" << endl; // Testing get

Map a;

Map b;

Map result;

result.insert("blah", 22.2);

result.insert("dope", 31);

a.insert("lmao", 55.3);

a.insert("yolo", 99.32);

a.insert("sma", 33234);

a.insert("haha", 34);

b.insert("YEET", 55.6);

b.insert("COOL", 8823.3);

cout << "This is a:" << endl; //Testing the inserts by printing

a.dump();

cout << "This is b:" << endl; //Testing the inserts by printing

b.dump();

a.erase("yolo"); //Testing erase

cout << "This is new a:" << endl; //Testing the erase by printing

a.dump();

a.swap(b); //Testing swap by swapping a and b

cout << "This is swapped a:" << endl;

a.dump();

cout << "This is swapped b:" << endl;

b.dump();//Testing the swap by printing

if (a.contains("YEET")) //Testing contains

{

cout << "contains is g so far" << endl;

}

if (!a.contains("idk man")) //Testing the false part of contains

{

cout << "contains is solid" << endl;

}

if (b.size() == 3)

cout << "size is solid" << endl; //Testing size function again

b.update("sma", 23.0); //Testing update

b.dump();

if (!b.update("ubby", 12)) //Testing update when key is not in list

cout << "update is solid" << endl;

a.insertOrUpdate("COOL", 13.0); //Testing insert/update when already in list (ipso facto should update)

a.insertOrUpdate("heyoo", 3.33); //Testing insert/update when not already in list (ipso facto should insert)

a.dump();

b.erase("lmao"); b.erase("haha"); b.erase("sma"); //Testing erasing all to ensure delete can delete single item lists

b.dump();

b.insert("sweet", 32.1);

b.insert("nice", 12.55);

b.insert("heyoo", 3.32);

b.insert("water", -3.0);

b.insert("YEET", 55.6); //Reinserting to test combined

cout << "a before combine is: " << endl;

a.dump();

cout << "b before combine is: " << endl;

b.dump();

if (combine(a, b, result)) // Combined testing with new map result

cout << "it is true!" << endl;

else

cout << "it is false!" << endl;

cout << "The result is: " << endl;

result.dump();

cout << "a before subtract is: " << endl;

a.dump();

cout << "b before subtract is: " << endl;

b.dump();

subtract(b, a, b);

cout << "The result is: " << endl;

b.dump();// Subtracted testing with aliases

}