**Project 2 Report**

The linked lists I designed in this project are circular double linked lists. After the end of the list, it goes back to the head of the list. Every node has prev and next pointer that points previous and next node. Node also has keyData and valueData. The first node is called the head, and the last pointer is the prev node of the head node, which also means the **next** pointer of the last node is the head of the list. The list also uses a dummy node for the head of the list.

**Pseudocode**

bool insert(const KeyType& key, const ValueType& value)

{

Inserts a new node with the parameter values of keyData and ValueData in increasing order of value. The loop finds a position to insert where it is smaller than the data at the position.

}

bool update(const KeyType& key, const ValueType& value)

{

Finds a node with same keyData through the loop, and changes ValueData of the node to parameter value.

}

bool insertOrUpdate(const KeyType& key, const ValueType& value)

{

Try inserting the parameter values first, and if insert function returns false, it uses update function, which means it tries to update the ValueData of the node that has same keyData with parameter key, it tries to insert a new node when there is no matching key.

}

bool erase(const KeyType& key)

{

Look for the node with same keyData as the parameter value through the loop, and deletes the node if it matches. Returns false if there is no matching key. After erasing it, it decrement size.

}

void swap(Map& other)

{

Creates a temporary pointer to swap the head pointer and creates a temporary integer pointers to swap the size variable.

}

bool merge(const Map& m1, const Map& m2, Map& result)

{

Merges two map through using insert function. If one of the Map is empty, it copies the entire map with the pair to the result. The function makes temporary map with same data from m1 and m2, and compares each data. If the key matches, but the valueData is different, it erases from the map. After this erasing process, it inserts the Datas to a result Map,

}

void reassign(const Map& m, Map& result)

{

It makes a temporary array, and keeps the value from m. The function inserts a key value pair with original key and value with 1 incremented index., The last key index contains the first value from the original map data.

}

**Tests**

Map m;

assert(m.empty()); //tests empty()

ValueType v = -1234.5;

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

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

assert(m.size() == 1); //test size(), check if it is inserted and size is incremented

KeyType k = "hello";

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

assert(m.contains("xyz")); //test contains(), check if it is inserted properly

Map n;

m.swap(n); //test swap()

KeyType a = "Bye";

ValueType b = 215;

assert(n.get(0, a, b) && a == "xyz" && b == 9876.5); //check if it’s swapped properly

Map mm;

mm.insert("Little Ricky", 3.206);

mm.insert("Ethel", 3.538);

mm.insert("Ricky", 3.350);

mm.insert("Lucy", 2.956);

mm.insert("Ethel", 3.538);

mm.insert("Fred", 2.956);

mm.insert("Lucy", 2.956);

assert(mm.size() == 5); // duplicate "Ethel" and "Lucy" were not added

string x;

double y;

mm.get(0, x, y);

assert(x == "Ethel"); // "Ethel" is greater than exactly 0 items in mm

mm.get(4, x, y);

assert(x == "Ricky"); // "Ricky" is greater than exactly 4 items in mm

mm.get(2, x, y);

assert(x == "Little Ricky"); // "Little Ricky" is greater than exactly 2 items in mm

assert(y == 3.206); // the value corresponding to the key "Little Ricky"

Map m1;

m1.insert("Fred", 2.956);

Map m2;

m2.insert("Ethel", 3.538);

m2.insert("Lucy", 2.956);

m1.swap(m2);

assert(m1.size() == 2 && m1.contains("Ethel") && m1.contains("Lucy") && m2.size() == 1 && m2.contains("Fred"));

m2.insert("Ethel", 3.734);

m2.insert("Lucy", 1.984);

Map m3;

assert(!merge(m1, m2, m3) && m3.contains("Fred") && !m3.contains("Lucy") & !m3.contains("Ethel")); //test merge() and see if wrong data not contained

Map m4;

reassign(m2, m4); //test reassign

Map gpas;

gpas.insert("Fred", 2.956);

assert(!gpas.contains(""));

gpas.insert("Ethel", 3.538);

gpas.insert("", 4.000);

gpas.insert("Lucy", 2.956);

assert(gpas.contains("")); //check if blank counts as a key

gpas.erase("Fred");

assert(gpas.size() == 3 && gpas.contains("Lucy") && gpas.contains("Ethel") && gpas.contains(""));

string kk;

double vv;

assert(gpas.get(1, kk, vv) && kk == "Ethel");

assert(gpas.get(0, kk, vv) && kk == "");

cout << "Passed all tests" << endl;