The constructor of Sequence constructs an empty Sequence.

head->nullptr

tail->nullptr

m\_size=0

When insert function is called, a new node is added to the empty linked list

head

Node’s content is like

|ItemType m\_value |

|int m\_count |

🡨|Node\* previous |

|Node\* next |🡪

When consecutive nodes are inserted into the linked list, each new node is added to the front.

m\_size is the sum of m\_count in all nodes.

Pseudocode

Constructor

Initialize private members

Destructor

Go through the linked list

Then store address of the next node to a variable

Delete the said node

Assign the address back to the pointer

Copy constructor

Initialize the head

Go through the linked list backwards starting from the tail

Assign other’s size to the size of the linked list

Assignment operator

Delete all the nodes

Assign the nullptr

Again, going backwards insert said RHS into the value for the x number of times

Assign rhs to this

Return the object of this

Insert function

If the linked list is empty

Add a new node to the front

Equalize the head and the tail

If the list is not empty

Find the node with the value using the find function implemented above

increase

Erase function

Find the node with the value that needs to be deleted

If it’s not at nullptr

If the node is not greater than 1, then we will need to decrease the size and return 1

If the size is equal to 1, we’ll have to delete the node

If it’s pointed at a nullptr, return 0

Remove function

Search the list

Assign 0 to nErssed

We now have to deal with 3 situations: depending on where the node in the linked list is at

If the node is only one in the linked list

Delete the node and decrease the size and then return size what was deleted.

If the node is in the first linked list, we’ll have to delete the node and decrease the size

If the node is the last in the linked list, we’ll again handle the pointers and delete the size and return the number of items deleted

If the node is somewhere in between the list, we’ll handle again said pointers and decrease the size and return the number deleted

Get function

Test whether i is inbound

Repeatedly

Get (i+1)th node

Return its value and count

addToFront function

create a new node

assign the value and count

let the node’s next pointer point to where head points to

if the linked list is not empty

make next node’s previous pointer point to new node

assign nulllptr to new node’s previous pointer

move head to new node

increase size and uniquesize

\*taken from professor nachenburg’s notes

find function

search the list to look for a node with value

if found

return the address

otherwise

return nullptr

combine function

assign content of ms1 to a template

repeatedly

get the value of each node into a temp

assign the number returned by get to n

repeatedly

insert temp to template for n times

assign the content of template to result

subtract function

assign content of ms1 to a template

repeatedly

get the value of each node into a temp

assign the number returned by get to n

repeatedly

erase the item with value equal to temp for n times from the template

assign the content of the template to result

Test cases

#include "Multiset.h"

#include <iostream>

#include <string>

#include <cassert>

using namespace std;

int main()

{

Sequence ms;

assert(xzx.empty()); // test the function "empty"

unsigned long x = 897;

assert(xzx.get(0, x) == 0 && x == 897); // test for function "get" when nothing in Sequence

assert(xzx.size() == 0 && xzx.uniqueSize()==0);

//test the function "insert"

xzx.insert(52);

xzx.insert(52);

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

assert(xzx.get(1, x) == 0 && x == 897);

assert(xzx.get(0, x) == 2 && x == 52); // test the function "get" after Multiset is inserted with values

// test the correctness of Sequence when a couple of values are inserted

assert(xzx.find(41) == 1 && xzx.find(52) == 1 && xzx.find(53) == 1 && xzx.find(40) == 0);

assert(xzx.find(41) == 1 && xzx.find(52) == 2 && xzx.find(53) == 3 && xzx.find(40) == 0);

// test the function "get" when several values coexist in Multiset

unsigned long a = 39;

assert((xzx.get(3, a) == 0 && a == 39) && (xzx.get(0, a) == 1 && a == 41) && (xzx.get(1, a) == 3 && a == 53) && (xzx.get(2, a) == 2 && a == 52) && (xzx.get(3, a) == 0 && a == 52));

// test the function "erase" and "eraseAll'

assert(xzx.erase(53) == 1 && xzx.erase(39) == 0);// test if there is no such item to be erased

assert(xzx.find(53) == 2);

assert(xzx.remove(53) == 2 && xzx.eraseAll(39) == 0);

assert(xzx.count(53) == 0);

Sequence z;

assert(z.erase(5) == 0);//test if there is no such item to be erased

assert(z.remoive(5) == 0);//test if there is no such item to be eraseAll

// test the function "swap"

Sequence ls;

assert(ls.empty());

xzx.swap(ls);

assert(ls.size() == 3 && xzx.size() == 0);

Sequence L;

assert(L.insert(10));

assert(L.insert(111));

assert(L.insert(111));

assert(L.insert(112));

assert(L.insert(1));

L.swap(L);// test an object swap with itself

L.dump();

// test for assignment operator

Sequence ns;

ns = ls;

// test for copy constructor

Sequence hs = ns;

Sequencea1;

a1.insert(52);

a1.insert(44);

interleave(hs, a1, ls);

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

}