Jordan Golan

405328335

Project 2 Report

**Description:**

I used a non-circular doubly linked list. I included both a head and tail pointer due to the fact that my list was non-circular. The head pointer, pointed to the first item in the list and the tail pointer pointed to the last item in the list. My code did not use a dummy node. The pointer to the previous node from the head node is nullptr, and the pointer to the next node from the tail node is nullptr. Within each list node is the data stored and a pointer to the next and previous nodes. The order of the linked list is just the order in which they were inserted.

Typical Set:

Tail

Head

Next

Nullptr

Next

Next

“Data 1”

“Data 3”

“Data 2”

Nullptr

Prev

Prev

Prev

Empty Set:

Head

Tail

Nullptr

**Pseudocode:**

Constructor:

* Set number of elements to zero
* Set head pointer to nullptr
* Set tail pointer to nullptr

Destructor:

* Create a temporary node
* Iterate through the list as long as head pointer is not pointing to nullptr.
  + Set temp node to head
  + Move head pointers to next node
  + Delete temp node

Copy Constructor:

* Set number of elements to other’s number of elements
* If the set is empty set the head and tail pointers equal to nullptr.
* Iterate through each node in the list so it copies all but the last node
  + In the iteration, make a new node
  + copy data from node in other
  + set previous and next pointer in copy
* copy the last node, and set the next to nullptr

Assignment Operator:

* check if this object isn’t being set to itself
  + if it’s not, copy the rhs to a temp variable
  + then swap temp

Insert:

* check to see if value is already in the set
* If set is empty
  + Create a new node
  + Insert the data value into the new node
  + New previous and next pointers to nullptr
  + Add 1 to number of elements
* If set is not empty
  + Create a new node
  + Insert data value into the new node at the end
  + Set the previous pointer to the old tail
  + Set next pointer to nullptr
  + Set tail to the new node
  + Add one to elements

Erase:

* Check if the set contains value, if not return false
* Iterate through the set to find value, and set pointer to that node
* Adjust head, tail, previous, and next pointers
* Delete the node containing value

Contains:

* Iterate through the list, to see if value is present
* Return true if it is found

Get:

* Check to see if index position value is valid
* Create a temporaroy copy of the set
* Delete max value from copy i-times
* Find the new max value, and set value equal to this new max

Swap:

* Swap number of elements in each set
* Swap the heads in each set
* Swap the tails in each set

**Tests:**

//Test unite

Set s1; // create new set

assert(s1.insert("hi"));// test unite

assert(s1.insert("hello"));// test unite

assert(s1.insert("yes"));// test unite

assert(s1.insert("no"));// test unite

assert(s1.insert("yee"));// test unite

Set s2; // create new set

Set result; // create new set

assert(s2.insert("hi")); // test unite

assert(s2.insert("naw")); // test unite

assert(s2.insert("hello"));// test unite

assert(s2.insert("please"));// test unite

assert(s2.insert("no"));// test unite

unite(s1, s2, result);// test unite

assert(result.contains("hi"));// test unite

assert(result.contains("hello"));// test unite

assert(result.contains("yes"));// test unite

assert(result.contains("no"));// test unite

assert(result.contains("yee"));// test unite

assert(result.contains("naw"));// test unite

assert(result.contains("please"));// test unite

//Test difference

difference(s1, s2, result);

assert(result.contains("naw"));

assert(result.contains("yee"));

assert(result.contains("yes"));

assert(!result.contains("hi"));// make sure it erased the correct member

Set test; // create new set

assert(test.size() == 0); // test size is zero when set is made

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

assert(test.insert("food")); // test insert, put element into list

assert(test.size() == 1); // test size function

assert(test.insert("pizza")); //test insert

assert(test.contains("food")); // test contains

assert(test.contains("pizza")); // test contains

assert(test.erase("food")); // test erase

assert(!test.contains("food")); // test erase and contains

string x;

assert(test.get(0, x) && x == "pizza"); // test get

assert(test.insert("dog")); // insert another item

assert(test.insert("cat")); // insert cat

Set copy(test); // test copy contructor

assert(copy.size() == 3); // tests size of copy

assert(copy.contains("dog")); // test if copy was created correctly, tests contents

assert(copy.erase("dog")); //erase from copy

assert(test.size() > copy.size()); // test sizes of both Sets

test.swap(copy); // test swap

assert(copy.size() > test.size()); // test if swap worked

assert(!test.contains("dog")); // further test of swap

assert(!test.empty()); // test empty when set is not empty

assert(copy.get(2,x) && x == "cat"); // test get

// test assignment operator

Set test2;

test2 = copy;

assert(test2.get(0,x) && x == "pizza");

assert(test2.size() == 3);