Brian Han-Wei Chang

UID: 405-142-813

CS 32 - Project 2 - Report

**A description of the design of your doubly-linked list implementation. (A couple of**

**sentences will probably suffice, perhaps with a picture of a typical Set and an empty Set. Is**

**the list circular? Does it have a dummy node? What's in your list nodes? Are they in any**

**particular order?)**

For my doubly-linked list implementation, I chose to go with a non-circular list without the use of dummy nodes. All my lists are initialised with nullptr’s for my head and tail pointers. Each node has a “m\_value”, “next”, and “prev” values. My lists are in order of increasing size (smallest to largest) and the sorting is done via the insert function.

**Pseudocode for non-trivial algorithms (e.g., subtract).**

For insert:

1. Check if value already exists in list, if som return
2. Check if list is empty, if so, add value.
3. If list is not empty
   1. Add to front if it’s smaller than head
   2. Add to rear if it’s larger than tail
   3. Add to middle by using a loop and stopping at the first node than our insertion value is greater than.

For all successful inserts, increment list size, redo the links, return true.

For erase:

1. Check if list is empty, if so, return.
2. Check if value is contained in list, if not, return.
3. If value is contained in list:
   1. If it’s the head, delete head node, make second node head.
   2. If it’s the tail, delete tail node, make second last node tail.
   3. If it’s in the middle, delete target node, redo the links.

For all successful erasures, decrement list size, ensure links are good, return true.

For unite:

1. Create copies of both S1 and S2 sets.
2. Create a compiled set and a placeholder, “value”.
3. Insert all of S1 into compiled.
4. Insert all of S2 into compiled
   1. Assuming your insert fxn is done right, it wont add duplicates.
5. Use assignment operator to set result set = compiled set.
6. Return result set.

For subtract:

1. Create copies of both S1 and S2 sets.
2. Create a compiled set and a placeholder, “value”.
3. Get each item of set S1.
4. If the item is not contained in set S2, insert into compiled set.
5. Use assignment operator to set result set = compiled set.
6. Return result set.

**A list of test cases that would thoroughly test the functions. Be sure to indicate the purpose**

**of the tests. For example, here's the beginning of a presentation in the form of code:**

The tests were performed on a set of strings (i.e., ItemType was a type alias for

std::string).

#include <iostream>

#include <string>

#include "Set.h"

**using** **namespace** std;

**int** main()

{

// string test routine

Set ss;

ss.insert("lavash");

ss.insert("roti");

ss.insert("chapati");

ss.insert("injera");

ss.insert("roti");

ss.insert("matzo");

ss.insert("injera");

assert(ss.size() == 5); // duplicate "roti" and "injera" were not added

string x;

ss.get(0, x);

assert(x == "chapati"); // "chapati" is greater than exactly 0 items in ss

ss.get(4, x);

assert(x == "roti"); // "roti" is greater than exactly 4 items in ss

ss.get(2, x);

assert(x == "lavash"); // "lavash" is greater than exactly 2 items in ss

assert(ss.erase("roti"));

assert(ss.size() == 4);

assert(!ss.empty());

Set gg;

gg.insert("dosa");

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

gg.insert("tortilla");

gg.insert("");

gg.insert("focaccia");

assert(gg.contains(""));

gg.erase("dosa");

assert(gg.size() == 3 && gg.contains("focaccia") && gg.contains("tortilla") && gg.contains(""));

string v;

assert(gg.get(1, v) && v == "focaccia");

assert(gg.get(0, v) && v == "");

assert(!gg.empty());

// test code for swap fxn

Set aa;

aa.insert("rice");

aa.insert("beans");

aa.insert("bread");

aa.insert("pasta");

Set bb;

bb.insert("chicken");

bb.insert("pork");

bb.insert("beef");

aa.swap(bb);

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

assert(bb.size()==4);

assert(aa.contains("chicken"));

assert(bb.contains("rice"));

assert(!aa.contains("pasta"));

assert(!bb.contains("beef"));

// //copy constructor test code

//

Set d(aa);

Set e(bb);

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

assert(e.size() == 4);

assert(d.contains("chicken"));

assert(e.contains("rice"));

assert(!d.contains("pasta"));

assert(!e.contains("beef"));

//Destructor and assignment operator test code.

Set abc;

abc.insert("Kobe");

abc.insert("Shaq");

abc.insert("Lebron");

Set xyz;

xyz.insert("Kareem");

xyz.insert("Magic");

xyz.**operator**=(abc);

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

assert(xyz.contains("Kobe"));

assert(xyz.contains("Shaq"));

assert(xyz.contains("Lebron"));

assert(!xyz.contains("Kareem"));

assert(!xyz.contains("Magic"));

//unite fxn test code

Set result;

result.insert("Sprouts");

result.insert("Okra");

result.insert("Asparagus");

Set bloo;

bloo.insert("apples");

bloo.insert("pears");

bloo.insert("melons");

bloo.insert("grapes");

bloo.insert("bananas");

Set bbo;

bbo.insert("oranges");

bbo.insert("pears");

bbo.insert("mangoes");

bbo.insert("grapes");

bbo.insert("lychees");

unite(bloo,bbo,result);

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

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

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

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

assert(!result.contains("Sprouts"));

assert(!result.contains("Asparagus"));

assert(result.size() == 8); //checking size

//Subtract fxn test code

Set jj;

jj.insert("1");

jj.insert("2");

jj.insert("3");

jj.insert("4");

jj.insert("5");

Set kk;

kk.insert("1");

kk.insert("6");

kk.insert("3");

kk.insert("8");

kk.insert("5");

subtract(jj, kk, result);

assert(result.contains("2"));

assert(result.contains("4"));

assert(!result.contains("1"));

assert(!result.contains("3"));

assert(!result.contains("5"));

cout << "All tests passed" << endl;

**return** (1);

}

//test routine for Unsigned longs.

**void** test()

{

Set uls;

assert(uls.insert(20));

assert(uls.insert(10));

assert(uls.size() == 2);

assert(uls.contains(10));

ItemType x = 30;

assert(uls.get(0, x) && x == 10);

assert(uls.get(1, x) && x == 20);

}

**int** main()

{

test();

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

}