CS32Project2 Report

* Q: 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?)

A: I used a circular doubly-linked-list with a dummy node in the front. A struct called Node is included that contains a pointer to a node called m\_next, which conceptually points to the next node in the list, a pointer to a node called m\_prev, which conceptually points to the previous node in the list, and a m\_value which stored the thing we are interested in each node. A m\_size variable which represent the number of nodes except the dummy node is also included in my Set.

* Q:[pseudocode](http://web.cs.ucla.edu/classes/winter23/cs32/pseudocode.html) for non-trivial algorithms (e.g., butNot).

A:

**Private function I defined:**

**int** Set::numOfGreater(**const** ItemType& val) **const**{

**int** num = 0;

**for** (Iterate through the linked-list){

**if**(the val is bigger than the node value){num++;}

}

**return** num;

}

**get function of set class:**

**bool** Set::get(**int** pos, ItemType& value) **const**{

**if** (pos is not rational) **return** **false**; // is the same as pos, assign that m\_value to

// value.

**for** (Iterate through every node of the linked-list){

**if**(numOfGreater(p->m\_value)is equal to the int from the input) {assign that node’s value to the value in the input and **return** **true**;}

}

**return** **false**;}

**but not function:**

**void** butNot(**const** Set& s1, **const** Set& s2, Set& result){

create a copy of s2 and assign result to s1

**for** (**int** i=0; i< s2copy.size(); i++){

ItemType v;

s2copy.get(i,v); <-for each values in s2

result.erase(v); <-erase that value in result if it contains it

}

}

* Q: 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:

A:

Test below conducted using ItemType = unsigned long.

#include "Set.h"

#include <iostream>

#include <cassert>

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

**void** test()

{

Set uls;

assert(uls.insert(20)); // check if insert function can properly work.

assert(uls.insert(10));

assert(!uls.empty()); // check if empty function work correctly.

assert(uls.size() == 2); // check the size function.

assert(uls.contains(10)); // check both cases of contains function

assert(!uls.contains(50));

ItemType x = 30;

assert(uls.get(0, x) && x == 10); // check if get function properly return value as desired

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

assert(!uls.get(4,x) && x == 20); // check if get function return false when pos is inappropriate

// and leave the x unchanged.

assert(uls.insert(30));

uls.dump();

Set uos;

assert(uos.insert(30));

assert(uos.insert(10));

assert(uos.insert(40));

assert(!uos.insert(10));

assert(!uos.erase(20)); //check if erase function return false when the argument is missing.

assert(uos.erase(10));

uos.dump(); // check if erase function correctly erase 10 from the list.

uos.swap(uls);

uos.dump(); // check if swap function correctly exchange elements of the two list.

Set result;

// unite (uls,uos,result);

// result.dump(); //check if unite functions as desired.(comment out when testing it)

butNot (uls,uos,result); // check if butNot fucntions as desired.(comment it when testing unite function)

result.dump();

}

**int** main()

{

test();

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

}

Note: default constructor, copy constructor, assignment operator and default destructor can be tested using other private and public functions in Set class. So, test cases for them are omitted. e.g. butNot can test copy constructor, which can check default constructor, and assignment operator. Swap can check default destructor…

Tests for string cases are similar.