Description:

A default-constructed Set has a head pointer, tail pointer, and a dummy node (with its next and previous pointers as NULL). The head and tail both point to the dummy node. The dummy node does not count as the first interesting node. The list is not circular, but linear. Nodes that are added contain a value, a pointer to the next node, and a pointer to the previous node.

Pseudocode:

Subtract:

Repeatedly:

Delete all nodes from result

Create a copy of s1

Create a copy of s2

Repeatedly:

if s1 value is not s2 value

insert value into result

else

skip to next s1 value to compare

Unite:

Repeatedly:

Delete all nodes from result

Create copy of s1

Create copy of s2

Repeatedly:

Insert s1 values into result

Repeatedly:

Insert s2 values into result

Test cases:

///// DEFAULT CONSTRUCTOR /////

Set s;

assert(s.size() == 0); //test size

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

assert(!s.erase("TEST")); //ensure nothing to erase

///// INSERTING /////

ItemType T = "A";

ItemType T2 = "B";

ItemType T3 = "C";

ItemType T4 = "D";

assert(s.insert(T)); //test ability to insert

assert(s.insert(T3));

assert(!s.insert(T)); //test inability to insert same value

///// GET /////

assert(s.get(0,T) && (T == "A")); //test actual inserted item

assert(s.get(1,T4) && (T4 == "C")); //test that value can change

///// CONTAINS /////

assert(s.contains(T)); //test contain function

assert(!s.contains(T2));

assert(s.contains(T3));

///// COPY CONSTRUCTOR /////

Set b(s); //intentional naming

assert(b.contains(T) && b.contains(T3)); //test copied elements

assert(b.size() == 2); //test copied size

///// ASSIGNMENT OPERATOR /////

Set d;

d = s; //test assignment operator compiles

assert(d.contains(T) && d.contains(T3)); //test actual values

assert(d.size() == 2); //test size

///// ERASING /////

assert(s.erase(T)); //test ability to erase

assert(s.size() == 1); //test actual erasing

assert(s.erase(T3));

assert(s.size() == 0);

///// SWAPPING /////

Set a;

Set c;

ItemType A = "R";

ItemType B = "T";

a.insert(A);

c.insert(B);

a.swap(c); //test swap

assert(a.contains(B)); //test values after swap

assert(c.contains(A));

assert(a.size() == 1 && c.size() == 1); //test size post-swap

///// UNITE /////

Set e;

unite(a,c,e); //test unite

assert(e.contains(A) && e.contains(B)); //test result

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

///// SUBTRACT /////

Set f;

f.insert(T);

f.insert(T2);

f.insert(T3);

Set g;

g.insert(T);

g.insert(T2);

Set h;

subtract(f,g,h); //test subtract

assert(h.contains(T3)); //test that unique values inserted

assert(!h.contains(T) && !h.contains(T2)); //test that shared values are not present