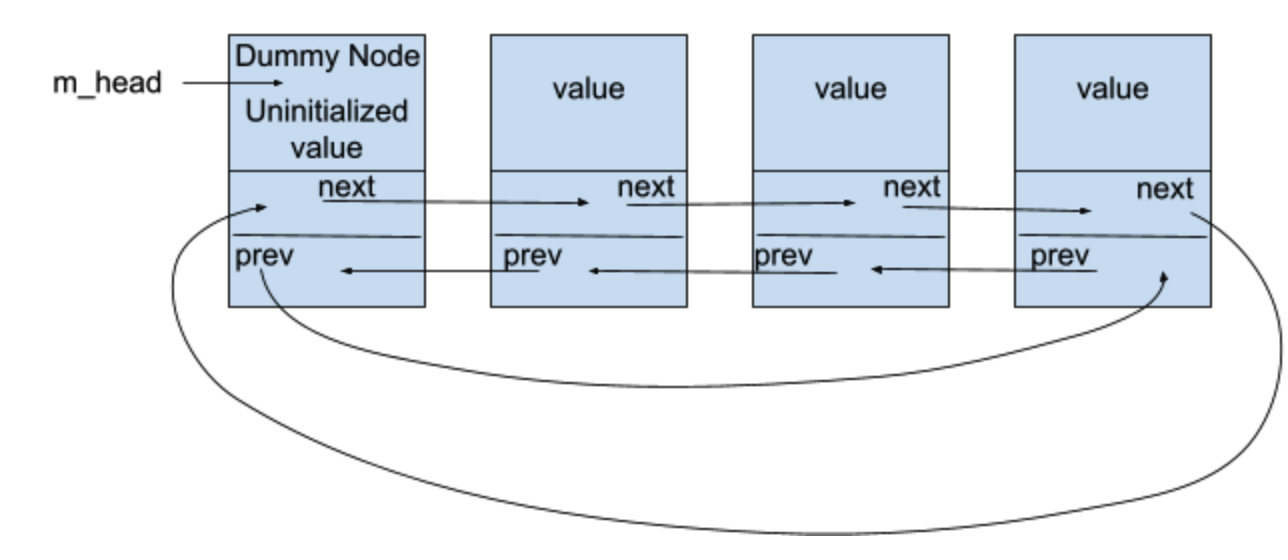
1. **Description of design**

I implemented a circularly doubly linked list with dummy node with data members m\_head and m\_size. Each node is composed of an ItemType value, previous pointer, and next pointer.



1. **Pseudocode**

Constructor:

Set size to 0 and create a dummy head node

Previous and next pointer points to itself

Copy Constructor:

Repeatedly:

Create new node

Set value of new node

Adjust new node’s previous pointer

New node becomes tail

Adjust new node’s next pointer to head

After set is copied, adjust head’s previous pointer and last node’s next pointer

Destructor:

Repeatedly, starting from tail:

Second element becomes the Node after tail (head)

Delete the node that head points to

Head now points to the next element

Delete remaining node

Assignment Operator:

If this set isn’t pointing to the same set as other,

Create temporary set from other;

And swap this set with temporary set

Insert function:

If value already exists, write false

Otherwise,

Create new node after tail

Assign value

New node becomes the tail

Adjust previous and next pointer of new tail

Increase size

Erase function:

If set is empty, write false

Otherwise,

If value is found:

Adjust node before and after to skip the node to be deleted.

Delete the node with the value

Decrement size

Contains function:

Repeatedly:

If value is found, write true

Get function:

If input is invalid, write false

Otherwise,

Repeatedly:

Compare current value with every other value in the list

If current value is less, increment counter

If at the end of the loop the current value is less than exactly i items,

Set parameter to the current value

Or write false if no items are less than exactly i items.

Swap function:

Swap size and head data member of this set and other

Unite function:

Create new set to assign later to result

Repeatedly:

Copy values in first set to new set

Repeatedly:

Copy values in second set to new set

(insert takes care of duplicates)

Result becomes new set

Difference function:

Create a new set to assign later to result

Repeatedly:

If current value in s1 is not contained in s2,

insert current value to new set

Repeatedly:

If current value in s2 is not contained in s1,

Insert current value to new set

Result becomes new set

1. **Test Cases**

Set a;

Set b;

assert(a.size() == 0 && b.size() == 0 && a.empty() && b.empty()); // properly initialized

a.insert("a");

a.insert("b");

a.insert("c");

b.insert("b");

b.insert("d");

a.erase("b");

assert(!a.contains("b")); // value b got erased

a.insert("b");

assert(!a.insert("a")); //test same item can't be inserted twice

assert(!b.insert("d")); // test same item can't be inserted twice

a.swap(b);

assert(a.size() == 2 && b.size() == 3); // a and b's size has been swapped

assert(a.contains("d") && b.contains("c")); // a and b's contents have been swapped

a = b; // assign b to a

assert(a.size() == 3 && b.size () == 3); // size is now the same

assert(a.contains("a") && a.contains("b") && !a.contains("d")); // a doesn't have its old content anymore, has b's content

a.erase("a");

a.erase("b");

a.erase("c");

assert(a.empty() && !a.erase("a")); // check set has been emptied, can't erase a value in empty array

assert(!a.contains("c") && b.contains("b")); // a does not have old contents anymore, b still has its value

assert(b.insert("d")); // insert new item into b

ItemType value = "d";

b.get(0, value);

assert(value == "d"); // value did not change

b.get(1, value);

assert(value == "c"); // value has changed

difference(b, b, b);

assert(b.empty()); // no difference, b is empty now

a.insert("a");

b.insert("b");

unite(b, b, b);

assert(b.size() == 1 && b.contains("b")); //make sure there are no duplicates

unite(b, a, b);

assert(b.size() == 2 && b.contains("a") && b.contains("b")); // Set a and b is merged, has values a and b

Set result;

assert(!b.insert("b"));//cannot insert same value

difference (a, b, result); // Set a has value a, Set b has value a and b, result should have b

assert(result.size() == 1 && result.contains("b")); // test difference function

Set c;

Set d;

c.insert("me");

c.insert("my");

c.insert("mine");

difference (c, c, c);

assert(c.empty()); // c is now empty because no difference

d.insert("hello");

difference (c, d, c);

assert(c.size() == 1 && c.contains("hello")); // c now has same contents as d

Set e;

assert(!e.erase("Value"));

e.swap(d);

assert(d.empty()); // d is now empty, swapped with e

assert(e.contains("hello") && e.size() == 1); // e has d's contents

assert(!e.erase("bye")); //cannot erase content that doesn't exist

difference(d, d, e);

assert(e.empty()); // e is empty now because no difference

ItemType hi;

assert(!e.get(1, hi) && !d.get(0, hi)); //cannot get anything from empty set

e.insert("ice cream");

e.insert("cake");

assert(e.get(0, hi) && hi == "ice cream" && !e.get(2, hi)); // check get function, cannot get value that is 2 items less in a size 2 set

Set f(e); // copy constructor

ItemType bye;

assert(f.get(0, bye) && bye == "ice cream" && f.size() == 2 );// check Set f has same contents as Set e