**Zihan Xia**

**205838465**

**A brief description of notable obstacles you overcame.**

**I had trouble figuring out the double linked list since there are both previous and after to keep track of, but I was able to solve it by drawing graphs and turning it into code step by step. Also, the find function got me confused for a while because there is not a straightforward way to loop through the deque. I ended up creating a function in deque that gives the class garage access to its head pointer.**

**A list of the test data that could be used to thoroughly test your functions, along with the reason for each test.**

**#include <iostream>**

**#include <string>**

**#include <cassert>**

**#include "DequeNode.h"**

**#include "DequeNode.cpp"**

**#include "Deque.h"**

**#include "Deque.cpp"**

**#include "Garage.h"**

**int main()**

**{**

**using namespace std;**

**using namespace cs32;**

**// Deque Tests**

**Deque< int > dInt;**

**int x;**

**Deque< std::string > dString;**

**std::string s;**

**dInt.deleteFront( );**

**dInt.deleteRear( );**

**x = 12;**

**assert( dInt.deleteItem( x ) == false );**

**assert( dInt.size( ) == 0 );**

**assert( dInt.get( -10, x ) == false );**

**assert( dInt.isEmpty( ) );**

**s = "Hello";**

**dString.addToFront( s );**

**s = "World!";**

**dString.addToRear( s );**

**s = "Howard";**

**assert( dString.deleteItem( s ) == false );**

**assert( dString.size( ) == 2 );**

**assert( dString.get( -10, s ) == false );**

**assert( dString.isEmpty( ) == false );**

**assert( dString.get( 0, s ) && s == "Hello" );**

**assert( dString.get( 1, s ) && s == "World!" );**

**dString.deleteFront( );**

**s = "Hello";**

**assert( dString.deleteItem( s ) == false );**

**assert( dString.size( ) == 1 );**

**assert( dString.get( 0, s ) && s == "World!" );**

**assert( dString.isEmpty( ) == false );**

**dString.deleteRear( );**

**s = "World!";**

**assert( dString.deleteItem( s ) == false );**

**assert( dString.size( ) == 0 );**

**assert( dString.get( 0, s ) == false );**

**assert( dString.isEmpty( ) );**

**// Garage Tests**

**Garage g;**

**Garage gSix( 6 );**

**assert( g.size( ) == 0 );**

**assert( g.capacity( ) == 10 );**

**assert( gSix.size( ) == 0 );**

**assert( gSix.capacity( ) == 6 );**

**gSix.tossIn( "shoes" );**

**gSix.tossIn( "shirts" );**

**gSix.tossIn( "books" );**

**assert( gSix.size( ) == 3 );**

**assert( gSix.capacity( ) == 6 );**

**gSix.tossIn( "suitcases" );**

**assert( gSix.tossIn( "tables" ) == false ); // nothing thrown away**

**assert( gSix.tossIn( "chairs" ) == false ); // nothing thrown away**

**assert( gSix.size( ) == 6 );**

**assert( gSix.capacity( ) == 6 );**

**assert( gSix.find( "shoes" ) );**

**assert( gSix.find( "shirts" ) );**

**assert( gSix.find( "books" ) );**

**assert( gSix.find( "suitcases" ) );**

**assert( gSix.find( "tables" ) );**

**assert( gSix.find( "chairs" ) );**

**assert( gSix.tossIn( "boxes" ) ); // shoes thrown away...**

**assert( gSix.size( ) == 6 );**

**assert( gSix.capacity( ) == 6 );**

**assert( gSix.find( "boxes" ) );**

**assert( gSix.find( "shoes" ) == false );**

**assert( gSix.find( "shirts" ) );**

**assert( gSix.find( "books" ) );**

**assert( gSix.find( "suitcases" ) );**

**assert( gSix.find( "tables" ) );**

**assert( gSix.find( "chairs" ) );**

**cout<<gSix.printItems()<<endl;**

**assert( gSix.tossOut( "shoes" ) == false ); // shoes not found...**

**assert( gSix.use( "shoes" ) == false ); // shoes not found...**

**assert( gSix.use( "tables" ) );**

**assert( gSix.tossIn( "records" ) ); // shirts thrown away...**

**assert( gSix.size( ) == 6 );**

**assert( gSix.capacity( ) == 6 );**

**assert( gSix.find( "records" ) );**

**assert( gSix.find( "tables" ) );**

**assert( gSix.find( "boxes" ) );**

**assert( gSix.find( "shoes" ) == false );**

**assert( gSix.find( "shirts" ) == false );**

**assert( gSix.find( "books" ) );**

**assert( gSix.find( "suitcases" ) );**

**assert( gSix.tossOut( "books" ) );**

**assert( gSix.tossOut( "records" ) );**

**assert( gSix.tossOut( "shoes" ) == false);**

**assert( gSix.size( ) == 4 );**

**assert( gSix.capacity( ) == 6 );**

**assert( gSix.find( "records" ) == false );**

**assert( gSix.find( "tables" ) );**

**assert( gSix.find( "chairs" ) );**

**assert( gSix.find( "boxes" ) );**

**assert( gSix.find( "shoes" ) == false );**

**assert( gSix.find( "shirts" ) == false );**

**assert( gSix.find( "books" ) == false );**

**assert( gSix.find( "suitcases" ) );**

**assert( gSix.tossOut( "tables" ) );**

**assert( gSix.tossOut( "boxes" ) );**

**assert( gSix.tossOut( "suitcases" ) );**

**assert( gSix.size( ) == 1 );**

**assert( gSix.capacity( ) == 6 );**

**assert( gSix.find( "records" ) == false );**

**assert( gSix.find( "tables" ) == false );**

**assert( gSix.find( "boxes" ) == false );**

**assert( gSix.find( "shoes" ) == false );**

**assert( gSix.find( "shirts" ) == false );**

**assert( gSix.find( "books" ) == false );**

**assert( gSix.find( "suitcases" ) == false );**

**assert( gSix.find( "chairs" ) );**

**cout << "all tests passed!" << endl;**

**// Example test case for Deque**

**Deque<int> dIntTest;**

**assert(dIntTest.isEmpty());**

**dIntTest.addToFront(10);**

**dIntTest.addToRear(20);**

**assert(dIntTest.size() == 2);**

**int frontValue, rearValue;**

**assert(dIntTest.get(0, frontValue) && frontValue == 10);**

**assert(dIntTest.get(1, rearValue) && rearValue == 20);**

**dIntTest.deleteFront();**

**dIntTest.deleteRear();**

**assert(dIntTest.isEmpty());**

**// Example test case for Garage**

**Garage testGarage(3);**

**testGarage.tossIn("Car");**

**testGarage.tossIn("Bike");**

**assert(testGarage.size() == 2);**

**testGarage.use("Car");**

**testGarage.tossIn("Scooter");**

**assert(testGarage.tossIn("Truck") == true); // "Bike" thrown away**

**assert(testGarage.find("Bike") == false);**

**assert(testGarage.find("Car") == true);**

**cout << "Additional tests passed!" << endl;**

**Deque< string > d; //delete functions on empty, size 1, and size 2 cases**

**d.deleteFront();**

**d.deleteRear();**

**d.deleteItem("a");**

**d.addToRear("first");**

**d.deleteFront();**

**d.addToFront("first");**

**d.addToRear("second");**

**d.deleteFront();**

**assert(d.front() == "second");**

**d.addToFront("first");**

**d.deleteRear();**

**assert(d.front() == "first");**

**d.addToRear("second");**

**d.deleteItem("first");**

**assert(d.front() == d.rear() && d.front() == "second");**

**d.addToFront("first");**

**d.deleteItem("second");**

**assert(d.front() == d.rear() && d.front() == "first");**

**d.deleteItem("first");**

**assert(d.isEmpty());**

**d.addToRear("first");**

**Deque<string > d2; //assignment operator**

**d2 = d;**

**d2.addToRear("second");**

**assert(d2.front() == "first");**

**d.addToFront("zero");**

**assert(d2.size() == 2);**

**assert(d2.front() == "first");**

**Garage d3(3); //Garage**

**d3.tossIn("first");**

**d3.tossOut("first");**

**assert(d3.size() == 0);**

**d3.tossIn("first");**

**d3.tossIn("second");**

**d3.tossIn("third");**

**assert(d3.tossIn("fourth") == true);**

**assert(d3.find("first") == false);**

**assert(d3.use("second") == true);**

**assert(d3.tossIn("first") == true);**

**assert(d3.find("third") == false);**

**cout << "More tests passed!" << endl;**

**return( 0 );**

**}**