Course: Principles of Software Design – ENSF 480

Lab 2

Instructor Name: M. Moussavi

Student Name: Gerardo Garcia de Leon

Lab Section B02

Date Submitted: September 23rd 2024

EXERCISE A:

```
/*
* File Name: dictionaryList.h
* Assignment: Lab 2 Exercise A
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 20, 2023
*/

#ifndef DICTIONARY_H
#define DICTIONARY_H
#include <iostream>
using namespace std;

// class DictionaryList: GENERAL CONCEPTS
//
//    key/datum pairs are ordered.  The first pair is the pair with
//    the lowest key, the second pair is the pair with the second
//    lowest key, and so on.  This implies that you must be able to
//    compare two keys with the < operator.
//
//    Each DictionaryList object has a "cursor" that is either attached
//    to a particular key/datum pair or is in an "off-list" state, not
//    attached to any key/datum pair.  If a DictionaryList is empty, the
//    cursor is automatically in the "off-list" state.

#include "mystring_B.h"

// Edit these typedefs to change the key or datum types, if necessary.
typedef int Key;
```

```
typedef Mystring Datum;

// THE NODE TYPE
//    In this exercise the node type is a class, that has a ctor.
//    Data members of Node are private, and class DictionaryList
//    is declared as a friend. For details on the friend keyword refer to your
//    lecture notes.

class Node {
  friend class DictionaryList;
private:
  Key keyM;
  Datum datumM;
  Node *nextM;

  // This ctor should be convenient in insert and copy operations.
  Node(const Key& keyA, const Datum& datumA, Node *nextA);
};

class DictionaryList {
public:
  DictionaryList();
  DictionaryList(const DictionaryList& source);
  DictionaryList& operator =(const DictionaryList& rhs);
  ~DictionaryList();

  int size() const;
  // PROMISES: Returns number of keys in the table.

  int cursor_ok() const;
```

```
// PROMISES:
//   Returns 1 if the cursor is attached to a key/datum pair,
//   and 0 if the cursor is in the off-list state.

const Key& cursor_key() const;
// REQUIRES: cursor_ok()
// PROMISES: Returns key of key/datum pair to which cursor is attached.

Datum& cursor_datum() const;
// REQUIRES: cursor_ok()
// PROMISES: Returns datum of key/datum pair to which cursor is attached.

void insert(const Key& keyA, const Datum& datumA);
// PROMISES:
//   If keyA matches a key in the table, the datum for that
//   key is set equal to datumA.
//   If keyA does not match an existing key, keyA and datumM are
//   used to create a new key/datum pair in the table.
//   In either case, the cursor goes to the off-list state.

void remove(const Key& keyA);
// PROMISES:
//   If keyA matches a key in the table, the corresponding
//   key/datum pair is removed from the table.
//   If keyA does not match an existing key, the table is unchanged.
//   In either case, the cursor goes to the off-list state.

void find(const Key& keyA);
// PROMISES:
//   If keyA matches a key in the table, the cursor is attached
```

```
  //   to the corresponding key/datum pair.
  //   If keyA does not match an existing key, the cursor is put in
  //   the off-list state.

  void go_to_first();
  // PROMISES: If size() > 0, cursor is moved to the first key/datum pair
  //   in the table.

  void step_fwd();
  // REQUIRES: cursor_ok()
  // PROMISES:
  //   If cursor is at the last key/datum pair in the list, cursor
  //   goes to the off-list state.
  //   Otherwise the cursor moves forward from one pair to the next.

  void make_empty();
  // PROMISES: size() == 0.

  friend ostream& operator <<(ostream& os, DictionaryList& rhs);
  // REQUIRES: rhs is a reference to a DictionaryList as a source
  // PROMISES: to overload the << operator to allow the desired output to be displayed

  const char* operator [](int index);
  // REQUIRES: list is a reference to a DictionaryList as a source and a valid index
  // PROMISES: to overload the [] operator to allow the desired output to be displayed

private:
  int sizeM;
  Node *headM;
  Node *cursorM;
```

```
  void destroy();
  // Deallocate all nodes, set headM to zero.

  void copy(const DictionaryList& source);
  // Establishes *this as a copy of source.  Cursor of *this will
  // point to the twin of whatever the source's cursor points to.

};

#endif
```

```
/*
* File Name: dictionaryList.cpp
* Assignment: Lab 2 Exercise A
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 20, 2023
*/
#include <assert.h>
#include <iostream>
#include <stdlib.h>
#include "dictionaryList.h"
#include "mystring_B.h"

using namespace std;

Node::Node(const Key& keyA, const Datum& datumA, Node *nextA)
  : keyM(keyA), datumM(datumA), nextM(nextA)
{
```

```
}

DictionaryList::DictionaryList()
  : sizeM(0), headM(0), cursorM(0)
{
}

DictionaryList::DictionaryList(const DictionaryList& source)
{
  copy(source);
}

DictionaryList& DictionaryList::operator =(const DictionaryList& rhs)
{
  if (this != &rhs) {
    destroy();
    copy(rhs);
  }
  return *this;
}

DictionaryList::~DictionaryList()
{
  destroy();
}

int DictionaryList::size() const
{
  return sizeM;
}
```

```
int DictionaryList::cursor_ok() const
{
  return cursorM != 0;
}

const Key& DictionaryList::cursor_key() const
{
  assert(cursor_ok());
  return cursorM->keyM;
}

Datum& DictionaryList::cursor_datum() const
{
  assert(cursor_ok());
  return cursorM->datumM;
}

void DictionaryList::insert(const int& keyA, const Mystring& datumA)
{
  // Add new node at head?
  if (headM == 0 || keyA < headM->keyM) {
    headM = new Node(keyA, datumA, headM);
    sizeM++;
  }

  // Overwrite datum at head?
  else if (keyA == headM->keyM)
    headM->datumM = datumA;
```

```
// Have to search ...
else {

  //POINT ONE

  // if key is found in list, just overwrite data;
  for (Node *p = headM; p !=0; p = p->nextM)
                {
                        if(keyA == p->keyM)
                        {
                                p->datumM = datumA;
                                return;
                        }
                }

  //OK, find place to insert new node ...
  Node *p = headM ->nextM;
  Node *prev = headM;

  while(p !=0 && keyA >p->keyM)
                {
                        prev = p;
                        p = p->nextM;
                }

  prev->nextM = new Node(keyA, datumA, p);
  sizeM++;
}
cursorM = NULL;
```

```
}

void DictionaryList::remove(const int& keyA)
{
    if (headM == 0 || keyA < headM -> keyM)
        return;

    Node *doomed_node = 0;

    if (keyA == headM-> keyM) {
        doomed_node = headM;
        headM = headM->nextM;

        // POINT TWO
    }
    else {
        Node *before = headM;
        Node *maybe_doomed = headM->nextM;
        while(maybe_doomed != 0 && keyA > maybe_doomed-> keyM) {
            before = maybe_doomed;
            maybe_doomed = maybe_doomed->nextM;
        }

        if (maybe_doomed != 0 && maybe_doomed->keyM == keyA) {
            doomed_node = maybe_doomed;
            before->nextM = maybe_doomed->nextM;
        }


    }
```

```
    if(doomed_node == cursorM)
        cursorM = 0;

    delete doomed_node;          // Does nothing if doomed_node == 0.
    sizeM--;
}

void DictionaryList::go_to_first()
{
    cursorM = headM;
}

void DictionaryList::step_fwd()
{
    assert(cursor_ok());
    cursorM = cursorM->nextM;
}

void DictionaryList::make_empty()
{
    destroy();
    sizeM = 0;
    cursorM = 0;
}

ostream& operator <<(ostream& os, DictionaryList& rhs) //This is the implementation of the
binary operator << as a non-member function
{
        rhs.go_to_first();
        while(rhs.cursor_ok()){
```

```
            os << rhs.cursor_key() << " " << rhs.cursor_datum() <<endl;
            rhs.step_fwd();
        }
        return os;
}


const char* DictionaryList::operator [](int index) //This is the implementation of the binary operator [] as a member function
{
        this->go_to_first();
        assert(index >= 0 && index < this->size());
        for (int i = index; i > 0; i--)
        {
            this->step_fwd();
        }

        return this->cursor_datum().c_str();
}

// The following function are supposed to be completed by the stuents, as part
// of the exercise B part II. the given fucntion are in fact place-holders for
// find, destroy and copy, in order to allow successful linking when you're
// testing insert and remove. Replace them with the definitions that work.

void DictionaryList::find(const Key& keyA)
{
        if(headM == 0){
            cout << "There is no list to search" << endl;
            exit(1);
```

```
        }


        for(Node* ptr = headM; ptr != 0; ptr = ptr ->nextM){
                if(keyA == ptr -> keyM){ //If keyA matches the value held by the current nodes's keyM, set cursorM to that node
                        cursorM = ptr;
                        return;
                }
        }
        cursorM = NULL; //We didn't find a key, so set to null
}



void DictionaryList::destroy()
{
  while(headM != 0){
          Node* temp = headM;
          headM = headM -> nextM;
          delete temp;
  }

  headM = 0; //headM should already be 0 after the while loop, but just to be safe we set it to zero
}



void DictionaryList::copy(const DictionaryList& source)
{
  if(source.headM == 0){
          this -> headM = 0;
          this -> cursorM = 0;
```

```
        this -> sizeM = 0;
        return;
  }

  this -> headM = new Node (source.headM -> keyM, source.headM -> datumM, nullptr);
//Creating the first node
  Node* next_src_node = source.headM -> nextM;
  Node* next_cpy_node = this -> headM;

  while(next_src_node != 0){ //As long as there's nodes to copy
        next_cpy_node -> nextM = new Node(next_src_node -> keyM, next_src_node ->
datumM, nullptr);
        next_cpy_node = next_cpy_node -> nextM;
        next_src_node = next_src_node -> nextM;
  }

  this -> sizeM = source.sizeM; //Copying the size

  if(source.cursorM == 0){ // If there is no cursor set, we set our copy's cursor to 0
        this -> cursorM = 0;
  }
  else{ // We use two iterators going at the same speed to stop when we find the source cursor.
I'm sure there is a more efficient way of doing this but I can't think of it
        Node* src_cursor = source.headM;
        Node* cpy_cursor = this -> headM;

        while(src_cursor != source.cursorM){
              src_cursor = src_cursor -> nextM;
              cpy_cursor = cpy_cursor -> nextM;
        }
        this -> cursorM = cpy_cursor;
```

```
  }
}

/*
* File Name: mystring_B.h
* Assignment: Lab 2 Exercise A
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 20, 2023
*/
#include <iostream>
#include <string>
using namespace std;

#ifndef MYSTRING_H
#define MYSTRING_H

class Mystring {

public:
  Mystring();
  // PROMISES: Empty string object is created.

  Mystring(int n);
  // PROMISES: Creates an empty string with a total capacity of n.
  //         In other words, dynamically allocates n elements for
  //         charsM,sets the lengthM to zero, and fills the first
  //         element of charsM with '\0'.

  Mystring(const char *s);
```

```
// REQUIRES: s points to first char of a built-in string.
// REQUIRES: Mystring object is created by copying chars from s.

~Mystring(); // destructor

Mystring(const Mystring& source); // copy constructor

Mystring& operator =(const Mystring& rhs); // assignment operator
// REQUIRES: rhs is reference to a Mystring as a source
// PROMISES: to make this-object (object that this is pointing to, as  a copy
//           of rhs.

bool operator >=(const Mystring& other) const;
// REQUIRES: other is reference to a Mystring as a source
// PROMISES: to overload the >= operator to allow the desired output to be displayed

friend ostream& operator<<(ostream& os, const Mystring& rhs);
// REQUIRES: rhs is a reference to a Mystring as a source
// PROMISES: to overload the << operator to allow the desired output to be displayed

bool operator <=(const Mystring& other) const;
// REQUIRES: other is a reference to a Mystring as a source
// PROMISES: to overload the <= operator to allow the desired output to be displayed

bool operator !=(const Mystring& other) const;
// REQUIRES: other is a reference to a Mystring as a source
// PROMISES: to overload the != operator to allow the desired output to be displayed

bool operator >(const Mystring& other) const;
// REQUIRES: other is a reference to a Mystring as a source
```

```
// PROMISES: to overload the > operator to allow the desired output to be displayed

bool operator <(const Mystring& other) const;
// REQUIRES: other is a reference to a Mystring as a source
// PROMISES: to overload the < operator to allow the desired output to be displayed

bool operator ==(const Mystring& other) const;
// REQUIRES: other is a reference to a Mystring as a source
// PROMISES: to overload the == operator to allow the desired output to be displayed

char& operator[](int index);
//REQUIRES: index to be a valid index in the range of this->charsM and rhs to be a reference to a Mystring as a source
//PROMISES: to return the character at index of charsM

int length() const;
// PROMISES: Return value is number of chars in charsM.

char get_char(int pos) const;
// REQUIRES: pos >= 0 && pos < length()
// PROMISES:
// Return value is char at position pos.
// (The first char in the charsM is at position 0.)

const char * c_str() const;
// PROMISES:
//   Return value points to first char in built-in string
//   containing the chars of the string object.

void set_char(int pos, char c);
```

```
 // REQUIRES: pos >= 0 && pos < length(), c != '\0'
 // PROMISES: Character at position pos is set equal to c.

 Mystring& append(const Mystring& other);

 // PROMISES: extends the size of charsM to allow concatenate other.charsM to
 //           to the end of charsM. For example if charsM points to "ABC", and
 //          other.charsM points to XYZ, extends charsM to "ABCXYZ".
 //

 void set_str(char* s);
 // REQUIRES: s is a valid C++ string of characters (a built-in string)
 // PROMISES:copys s into charsM, if the length of s is less than or equal lengthM.
 //          Othrewise, extends the size of the charsM to s.lengthM+1, and copies
 //          s into the charsM.



 int isEqual (const Mystring& s)const;
 // REQUIRES: s refers to an object of class Mystring
 // PROMISES: retruns true if charsM equal s.charsM.


private:

 int lengthM; // the string length - number of characters excluding \0
 char* charsM; // a pointer to the beginning of an array of characters, allocated dynamically.
 void memory_check(char* s);
 // PROMISES: if s points to NULL terminates the program.
};
```

```
#endif

/*
* File Name: mystring_B (1).cpp
* Assignment: Lab 2 Exercise A
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 20, 2023
*/
#include "mystring_B.h"
#include <string.h>
#include <iostream>
#include <cassert>
using namespace std;

Mystring::Mystring()
{
  charsM = new char[1];

  // make sure memory is allocated.
  memory_check(charsM);
  charsM[0] = '\0';
  lengthM = 0;
}

Mystring::Mystring(const char *s)
  : lengthM(strlen(s))
{
  charsM = new char[lengthM + 1];
```

```
 // make sure memory is allocated.
  memory_check(charsM);

  strcpy(charsM, s);
}

Mystring::Mystring(int n)
  : lengthM(0), charsM(new char[n])
{
  // make sure memory is allocated.
  memory_check(charsM);
  charsM[0] = '\0';
}

Mystring::Mystring(const Mystring& source):
  lengthM(source.lengthM), charsM(new char[source.lengthM+1])
{
  memory_check(charsM);
  strcpy (charsM, source.charsM);
}

Mystring::~Mystring()
{
  delete [] charsM;
}

int Mystring::length() const
{
  return lengthM;
}
```

```
char Mystring::get_char(int pos) const
{
  if(pos < 0 && pos >= length()){
    cerr << "\nERROR: get_char: the position is out of boundary." ;
  }

  return charsM[pos];
}

const char * Mystring::c_str() const
{
  return charsM;
}

void Mystring::set_char(int pos, char c)
{
  if(pos < 0 && pos >= length()){
    cerr << "\nset_char: the position is out of boundary."
         << " Nothing was changed.";
    return;
  }

  if (c != '\0'){
    cerr << "\nset_char: char c is empty."
         << " Nothing was changed.";
    return;
  }

  charsM[pos] = c;
```

```
}

Mystring& Mystring::operator =(const Mystring& S)
{
  if(this == &S)
    return *this;
  delete [] charsM;
  lengthM = (int)strlen(S.charsM);
  charsM = new char [lengthM+1];
  memory_check(charsM);
  strcpy(charsM,S.charsM);

  return *this;
}

bool Mystring::operator >=(const Mystring& other) const //This is the implementation of the
binary >= operator as a member function
{
        return this-> charsM >= other.charsM;
}

ostream& operator<<(ostream& os, const Mystring& S) //This is the implementation of the binary
<< operator as a non-member function
{
        os << S.charsM;
        return os;
}

bool Mystring::operator <=(const Mystring& other) const // This is the implementation of the
binary <= operator as a member function
{
```

```
	return this->charsM <= other.charsM;
}

bool Mystring::operator !=(const Mystring& other) const //This is the implementation of the binary != operator as a member function
{
	return this->charsM != other.charsM;
}

bool Mystring::operator >(const Mystring& other) const //This is the implementation of the binary != operator as a member function
{
	return this->charsM > other.charsM;
}

bool Mystring::operator <(const Mystring& other) const //This is the implementation of the binary != operator as a member function
{
	return this->charsM < other.charsM;
}

bool Mystring::operator ==(const Mystring& other) const //This is the implementation of the binary != operator as a member function
{
	return this->charsM == other.charsM;
}

char& Mystring::operator[](int index) // This is the implementation of the binary operator [] as a member function
{
	assert(index >= 0 && index < (int)strlen(charsM));
	return charsM[index];
```

```
}

Mystring& Mystring::append(const Mystring& other)
{
  char *tmp = new char [lengthM + other.lengthM + 1];
  memory_check(tmp);
  lengthM+=other.lengthM;
  strcpy(tmp, charsM);
  strcat(tmp, other.charsM);
  delete []charsM;
  charsM = tmp;

  return *this;
}

 void Mystring::set_str(char* s)
{
    delete []charsM;
    lengthM = (int)strlen(s);
    charsM=new char[lengthM+1];
    memory_check(charsM);

    strcpy(charsM, s);
}

int Mystring::isEqual (const Mystring& s)const
{
  return (strcmp(charsM, s.charsM)== 0);
}
```

```
void Mystring::memory_check(char* s)
{
  if(s == 0)
    {
      cerr <<"Memory not available.";
      exit(1);
    }
}

/*
* File Name: exBmain (1).cpp
* Assignment: Lab 2 Exercise A
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 20, 2023
*/
#include <assert.h>
#include <iostream>
#include "dictionaryList.h"

using namespace std;

DictionaryList dictionary_tests();

void test_copying();

void print(DictionaryList& dl);

void test_finding(DictionaryList& dl);
```

```
void test_operator_overloading(DictionaryList& dl);

int main()
{
  DictionaryList dl = dictionary_tests();

  test_copying();

 // Uncomment the call to test_copying when DictionaryList::copy is properly defined
 // test_finding(dl);
  test_operator_overloading(dl);

  return 0;
}

DictionaryList dictionary_tests()
{

  DictionaryList dl;

  assert(dl.size() == 0);
  cout << "\nPrinting list just after its creation ...\n";
  print(dl);

  // Insert using new keys.
  dl.insert(8001,"Dilbert");
  dl.insert(8002,"Alice");
  dl.insert(8003,"Wally");
  assert(dl.size() == 3);
```

```
  cout << "\nPrinting list after inserting 3 new keys ...\n";
  print(dl);
  dl.remove(8002);
  dl.remove(8001);
  dl.insert(8004,"PointyHair");
  assert(dl.size() == 2);
  cout << "\nPrinting list after removing two keys and inserting PointyHair ...\n";
  print(dl);

  // Insert using existing key.
  dl.insert(8003,"Sam");
  assert(dl.size() == 2);
  cout << "\nPrinting list after changing data for one of the keys ...\n";
  print(dl);

  dl.insert(8001,"Allen");
  dl.insert(8002,"Peter");
  assert(dl.size() == 4);
  cout << "\nPrinting list after inserting 2 more keys ...\n";
  print(dl);

  cout << "***----Finished dictionary tests----------***\n\n";
  return dl;
}

void test_copying()
{
   DictionaryList one;

  // Copy an empty list.
```

```
DictionaryList two;
assert(two.size() == 0);

// Copy a list with three entries and a valid cursor.
one.insert(319,"Randomness");
one.insert(315,"Shocks");
one.insert(335,"ParseErrors");
one.go_to_first();
one.step_fwd();

DictionaryList three(one);

assert(three.cursor_datum().isEqual("Randomness"));
one.remove(335);

cout << "Printing list--keys should be 315, 319\n";
print(one);

cout << "Printing list--keys should be 315, 319, 335\n";
print(three);

// Assignment operator check.
one = two = three = three;
one.remove(319);
two.remove(315);

cout << "Printing list--keys should be 315, 335\n";
print(one);

cout << "Printing list--keys should be 319, 335\n";
```

```
  print(two);

  cout << "Printing list--keys should be 315, 319, 335\n";
  print(three);

  cout << "***----Finished tests of copying----------***\n\n";
}

void print(DictionaryList& dl)
{
  if (dl.size() == 0)
    cout << "  List is EMPTY.\n";
  for (dl.go_to_first(); dl.cursor_ok(); dl.step_fwd()) {
    cout << "  " << dl.cursor_key();
    cout << "  " << dl.cursor_datum().c_str() << '\n';
  }
}

void test_finding(DictionaryList& dl)
{

     // Pretend that a user is trying to look up names.
     cout << "\nLet's look up some names ...\n";

     dl.find(8001);
     if (dl.cursor_ok())
        cout << "  name for 8001 is: " << dl.cursor_datum().c_str() << ".\n";
     else
        cout << "  Sorry, I couldn't find 8001 in the list. \n" ;
```

```
    dl.find(8000);
    if (dl.cursor_ok())
      cout << "  name for 8000 is: " << dl.cursor_datum().c_str() << ".\n";
    else
      cout << "  Sorry, I couldn't find 8000 in the list. \n" ;

    dl.find(8002);
    if (dl.cursor_ok())
      cout << "  name for 8002 is: " << dl.cursor_datum().c_str() << ".\n";
    else
      cout << "  Sorry, I couldn't find 8002 in the list. \n" ;

    dl.find(8004);
    if (dl.cursor_ok())
      cout << "  name for 8004 is: " << dl.cursor_datum().c_str() << ".\n";
    else
      cout << "  Sorry, I couldn't find 8004 in the list. \n" ;

   cout << "***----Finished tests of finding ------------------------***\n\n";
}
#if 1
void test_operator_overloading(DictionaryList& dl)
{

   DictionaryList dl2 = dl;
   dl.go_to_first();
   dl.step_fwd();
   dl2.go_to_first();

   cout << "\nTesting a few comparison and insertion operators." << endl;
```

```
// Needs to overload >= and << (insertion operator) in class Mystring
if(dl.cursor_datum() >= (dl2.cursor_datum()))
  cout << endl << dl.cursor_datum() << " is greater than or equal " << dl2.cursor_datum();
else
  cout << endl << dl2.cursor_datum() << " is greater than " << dl.cursor_datum();

// Needs to overload <= for Mystring
if(dl.cursor_datum() <= (dl2.cursor_datum()))
   cout << dl.cursor_datum() << " is less than or equal" << dl2.cursor_datum();
else
   cout << endl << dl2.cursor_datum() << " is less than " << dl.cursor_datum();

if(dl.cursor_datum() != (dl2.cursor_datum()))
   cout << endl << dl.cursor_datum() << " is not equal to " << dl2.cursor_datum();
else
   cout << endl << dl2.cursor_datum() << " is equal to " << dl.cursor_datum();


if(dl.cursor_datum() > (dl2.cursor_datum()))
   cout << endl << dl.cursor_datum() << " is greater than " << dl2.cursor_datum();
else
   cout << endl << dl.cursor_datum() << " is not greater than " << dl2.cursor_datum();

if(dl.cursor_datum() < (dl2.cursor_datum()))
   cout << endl << dl.cursor_datum() << " is less than " << dl2.cursor_datum();
else
   cout << endl << dl.cursor_datum() << " is not less than " << dl2.cursor_datum();
if(dl.cursor_datum() == (dl2.cursor_datum()))
   cout << endl << dl.cursor_datum() << " is equal to " << dl2.cursor_datum();
```

```
 else
     cout << endl << dl.cursor_datum() << " is not equal to " << dl2.cursor_datum();
cout << endl << "\nUsing square bracket [] to access elements of Mystring objects. ";

 char c = dl.cursor_datum()[1];
 cout << endl << "The socond element of "  << dl.cursor_datum() << " is: " << c;

 dl.cursor_datum()[1] = 'o';
 c = dl.cursor_datum()[1];
 cout << endl << "The socond element of "  << dl.cursor_datum() << " is: " << c;

 cout << endl << "\nUsing << to display key/datum pairs in a Dictionary list: \n";
 /* The following line is expected to display the content of the linked list
  * dl2 -- key/datum pairs. It should display:
  *   8001  Allen
  *   8002  Peter
  *   8003  Sam
  *   8004  PointyHair
  */
 cout << dl2;

 cout << endl << "\nUsing [] to display the datum only: \n";
 /* The following line is expected to display the content of the linked list
  * dl2 -- datum. It should display:
  *   Allen
  *   Peter
  *   Sam
  *   PointyHair
  */
```

```
    for(int i =0; i < dl2.size(); i++)
        cout << dl2[i] << endl;

    cout << endl << "\nUsing [] to display sequence of charaters in a datum: \n";
    /* The following line is expected to display the characters in the first node
     * of the dictionary. It should display:
     *   A
     *   l
     *   l
     *   e
     *   n
     */
    cout << dl2[0][0] << endl;
    cout << dl2[0][1] << endl;
    cout << dl2[0][2] << endl;
    cout << dl2[0][3] << endl;
    cout << dl2[0][4] << endl;

    cout << "\n\n***----Finished tests for overloading operators ----------***\n\n";
}
#endif
```

OUTPUT:

```
Printing list just after its creation ...
  List is EMPTY.

Printing list after inserting 3 new keys ...
  8001  Dilbert
  8002  Alice
```

8003 Wally

Printing list after removing two keys and inserting PointyHair ...

8003 Wally

8004 PointyHair

Printing list after changing data for one of the keys ...

8003 Sam

8004 PointyHair

Printing list after inserting 2 more keys ...

8001 Allen

8002 Peter

8003 Sam

8004 PointyHair

----Finished dictionary tests----------

Printing list--keys should be 315, 319

315 Shocks

319 Randomness

Printing list--keys should be 315, 319, 335

315 Shocks

319 Randomness

335 ParseErrors

Printing list--keys should be 315, 335

315 Shocks

335 ParseErrors

Printing list--keys should be 319, 335

319 Randomness

335 ParseErrors

Printing list--keys should be 315, 319, 335

315 Shocks

319 Randomness

335 ParseErrors

----Finished tests of copying----------

Testing a few comparison and insertion operators.

Peter is greater than or equal Allen

Allen is less than Peter

Peter is not equal to Allen

Peter is greater than Allen

Peter is not less than Allen

Peter is not equal to Allen

Using square bracket [] to access elements of Mystring objects.

The socond element of Peter is: e

The socond element of Poter is: o

Using << to display key/datum pairs in a Dictionary list:

8001 Allen

8002 Peter

8003 Sam

8004 PointyHair

Using [] to display the datum only:

Allen

Peter

Sam

PointyHair

Using [] to display sequence of charaters in a datum:

A

l

l

e

n

----Finished tests for overloading operators ----------

EXERCISE B:

```
/*
* File Name: point.h
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#ifndef POINT_H
#define POINT_H
#include <iostream>
using namespace std;

class Point{
```

```
    protected:
      static int count;
      int id;
      double x;
      double y;

    public:
      Point(double x, double y);
      void display() const;
      static int counter();
      static double distance(Point& first, Point& other);
      double distance(Point& other) const;
      int get_id() const;
      double get_x() const;
      double get_y() const;
      void set_x(double x);
      void set_y(double y);
};

#endif

/*
* File Name: point.cpp
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#include <iostream>
```

```
#include <cmath>
#include "point.h"
using namespace std;


int Point::count = 0;

Point::Point(double x, double y)
{
	id = 1001 + count;
	set_x(x);
	set_y(y);
	count++;
}

void Point::display() const
{
	cout << "\nX-coordinate: " << get_x();
	cout << "\nY-coordinate: " << get_y();
}

double Point::get_x() const
{
	return x;
}

double Point::get_y() const
{
	return y;
}
```

```
int Point::get_id() const
{
	return id;
}

void Point::set_y(double y)
{
	this->y = y;
}

void Point::set_x(double x)
{
	this->x = x;
}

int Point::counter()
{
	return count;
}

double Point::distance(Point& first, Point& other)
{
	return sqrt(pow((first.get_x() - other.get_x()), 2) + pow((first.get_y() - other.get_y()), 2));
}

double Point::distance(Point& other) const
{
	return sqrt(pow(((x) - (other.get_x())), 2) + pow(((y) - (other.get_y())), 2));
}
```

```
/*
* File Name: shape.h
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#ifndef SHAPE_H
#define SHAPE_H
#include <iostream>
#include <cstring>
#include <cmath>
#include "point.h"
using namespace std;

class Shape
{
        protected:
          Point origin;
          char* shapeName;

        public:
          Shape(double x, double y, const char* name);
          ~Shape();
          Shape(const Shape& src);
          const Point& get_Origin() const;
          const char* get_name() const;
          void display() const;
          double distance (Shape& other) const;
```

```
        static double distance (Shape& the_shape, Shape& other);
        void move (double dx, double dy);
        Shape& operator =(Shape& s);

};

#endif

/*
* File Name: shape.cpp
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#include <iostream>
#include <cstring>
#include <cmath>
#include "shape.h"
using namespace std;

Shape::Shape(double x, double y, const char* name) : origin(x, y)
{
        shapeName = new char[strlen(name) + 1];
        strcpy(shapeName, name);
}

Shape::~Shape()
{
```

```
	delete[] shapeName;
}

const Point& Shape::get_Origin() const
{
	return origin;
}

const char* Shape::get_name() const
{
	return shapeName;
}

void Shape::display() const
{
	cout << "Shape Name: " << shapeName <<endl;
	cout << "X-Coordinate: " << origin.get_x() <<endl;
	cout << "Y-Coordinate: " << origin.get_y() <<endl;
}

double Shape::distance(Shape& other) const
{
	return sqrt(pow(((origin.get_x()) - (other.origin.get_x())), 2) + pow(((origin.get_y()) - (other.origin.get_y())), 2));
}

double Shape::distance(Shape& the_shape, Shape& other)
{
	return sqrt(pow(((the_shape.origin.get_x()) - (other.origin.get_x())), 2) + pow(((the_shape.origin.get_y()) - (other.origin.get_y())), 2));
}
```

```
void Shape::move(double dx, double dy)
{
	origin.set_x(origin.get_x() + dx);
	origin.set_y(origin.get_y() + dy);
}

Shape& Shape::operator =(Shape& s)
{
	if(this != &s)
	{
		delete[] shapeName;
		this->origin = s.get_Origin();
		this->shapeName = new char[strlen(s.get_name()) + 1];
		strcpy(this->shapeName, s.get_name());
	}

	return *this;
}

Shape::Shape(const Shape& src):origin(src.origin)
{
	shapeName = new char[strlen(src.shapeName) + 1];
	strcpy(shapeName, src.get_name());
}

/*
* File Name: square.h
* Assignment: Lab 2 Exercise B
* Lab section: B02
```

*** Completed by: Gerardo Garcia de Leon**

*** Development Date: Sept 22, 2023**

***/**

```
#ifndef SQUARE_H
#define SQUARE_H

#include <iostream>
#include <cmath>
#include "shape.h"
using namespace std;

class Square: public Shape
{
        protected:
          double side_a;

        public:
          Square(double x, double y, double side, const char* name);
          double get_area() const;
          double perimeter() const;
          double get_sideA() const;
          void set_side_a(double side);
          void display() const;
};

#endif
```

/*

*** File Name: square.cpp**

```
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#include <iostream>
#include <cmath>
#include "square.h"
using namespace std;

Square::Square(double x, double y, double side, const char* name): Shape(x, y, name)
{
	side_a = side;
}

double Square::get_area() const
{
	return pow(side_a, 2);
}

double Square::perimeter() const
{
	return (4*side_a);
}

double Square::get_sideA() const
{
	return side_a;
}
```

```
void Square::set_side_a(double side)
{
	side_a = side;
}

void Square::display() const
{
	cout << "Square Name: " << Shape::get_name() << endl;
	cout << "X-Coordinate: " << origin.get_x() << endl;
	cout << "Y-Coordinate: " << origin.get_y() <<endl;
	cout << "Side a: " << get_sideA() << endl;
	cout << "Area: " << get_area() <<endl;
	cout << "Perimeter: " << perimeter() << endl;
}

/*
* File Name: rectangle.h
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#ifndef RECTANGLE_H
#define RECTANGLE_H

#include <iostream>
#include "square.h"
#include "shape.h"
```

```
using namespace std;

class Rectangle : public Square
{
        private:
          double side_b;

        public:
          Rectangle (double x, double y, double side1, double side2, const char* name);
          double get_area() const;
          double perimeter() const;
          double get_sideB() const;
          void set_side_b(double side);
          void display() const;
};

#endif

/*
* File Name: rectangle.cpp
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#include <iostream>
#include "rectangle.h"
using namespace std;
```

```
Rectangle::Rectangle(double x, double y, double side1, double side2, const char* name):
Square(x, y, side1, name)
{
	side_b = side2;
}

double Rectangle::get_area() const
{
	return side_b * get_sideA();
}

double Rectangle::perimeter() const
{
	return ((2*get_sideA()) + (2*side_b));
}

double Rectangle::get_sideB() const
{
	return side_b;
}

void Rectangle::set_side_b(double side)
{
	side_b = side;
}

void Rectangle::display() const
{
	cout << "Rectangle Name: " << Shape::get_name() << endl;
	cout << "X-Coordinate: " << origin.get_x() << endl;
```

```
        cout << "Y-Coordinate: " << origin.get_y() <<endl;
        cout << "Side a: " << Square::get_sideA() << endl;
        cout << "Side b: " << get_sideB() <<endl;
        cout << "Area: " << get_area() <<endl;
        cout << "Perimeter: " << perimeter() << endl;
}

/*
* File Name: graphicsWorld.cpp
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#include <iostream>
#include "graphicsWorld.h"
using namespace std;

void GraphicsWorld::run(){

        #if 1 // Change 0 to 1 to test Point
        Point m (6, 8);
        Point n (6,8);
        n.set_x(9);
        cout << "\nExpected to dispaly the distance between m and n is: 3";
        cout << "\nThe distance between m and n is: " << m.distance(n);
        cout << "\nExpected second version of the distance function also print: 3";
        cout << "\nThe distance between m and n is again: "
        << Point::distance(m, n);
```

```
#endif // end of block to test Point

#if 1 // Change 0 to 1 to test Square
cout << "\n\nTesting Functions in class Square:" <<endl;
Square s(5, 7, 12, "SQUARE - S");
s.display();
#endif // end of block to test Square

#if 1 // Change 0 to 1 to test Rectangle
cout << "\nTesting Functions in class Rectangle:" <<endl;
Rectangle a(5, 7, 12, 15, "RECTANGLE A");
a.display();
Rectangle b(16 , 7, 8, 9, "RECTANGLE B");
b.display();
double d = a.distance(b);
cout <<"\nDistance between square a, and b is: " << d << endl;
Rectangle rec1 = a;
rec1.display();
cout << "\nTesting assignment operator in class Rectangle:" <<endl;
Rectangle rec2 (3, 4, 11, 7, "RECTANGLE rec2");
rec2.display();
rec2 = a;
a.set_side_b(200);
a.set_side_a(100);
cout << "\nExpected to display the following values for objec rec2: " << endl;
cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n" << "Y-coordinate:
7\n"
<< "Side a: 12\n" << "Side b: 15\n" << "Area: 180\n" << "Perimeter: 54\n" ;
cout << "\nIf it doesn't there is a problem with your assignment operator.\n" << endl;
rec2.display();
```

```
	cout << "\nTesting copy constructor in class Rectangle:" <<endl;
	Rectangle rec3 (a);
	rec3.display();
	a.set_side_b(300);
	a.set_side_a(400);
	cout << "\nExpected to display the following values for objec rec2: " << endl;
	cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n" << "Y-coordinate:
7\n"
	<< "Side a: 100\n" << "Side b: 200\n" << "Area: 20000\n" << "Perimeter: 600\n" ;
	cout << "\nIf it doesn't there is a problem with your assignment operator.\n" << endl;
	rec3.display();
	#endif // end of block to test Rectangle

	#if 0 // Change 0 to 1 to test using array of pointer and polymorphism
	cout << "\nTesting array of pointers and polymorphism:" <<endl;
	Shape* sh[4];
	sh[0] = &s;
	sh[1] = &b;
	sh [2] = &rec1;
	sh [3] = &rec3;
	sh [0]->display();
	sh [1]->display();
	sh [2]->display();
	sh [3]->display();
	#endif // end of block to test array of pointer and polymorphism
}

int main(void)
{
	GraphicsWorld app;
```

```
	app.run();

	return 0;
}

/*
* File Name: graphicsWorld.h
* Assignment: Lab 2 Exercise B
* Lab section: B02
* Completed by: Gerardo Garcia de Leon
* Development Date: Sept 22, 2023
*/

#ifndef GRAPHICSWORLD_H
#define GRAPHICSWORLD_H

#include <iostream>
#include "shape.h"
#include "point.h"
#include "square.h"
#include "rectangle.h"
using namespace std;

class GraphicsWorld
{
	public:
	  void run();
};

#endif
```

OUTPUT:

Expected to dispaly the distance between m and n is: 3

The distance between m and n is: 3

Expected second version of the distance function also print: 3

The distance between m and n is again: 3

Testing Functions in class Square:

Square Name: SQUARE - S

X-Coordinate: 5

Y-Coordinate: 7

Side a: 12

Area: 144

Perimeter: 48

Testing Functions in class Rectangle:

Rectangle Name: RECTANGLE A

X-Coordinate: 5

Y-Coordinate: 7

Side a: 12

Side b: 15

Area: 180

Perimeter: 54

Rectangle Name: RECTANGLE B

X-Coordinate: 16

Y-Coordinate: 7

Side a: 8

Side b: 9

Area: 72

Perimeter: 34

Distance between square a, and b is: 11

Rectangle Name: RECTANGLE A

X-Coordinate: 5

Y-Coordinate: 7

Side a: 12

Side b: 15

Area: 180

Perimeter: 54

Testing assignment operator in class Rectangle:

Rectangle Name: RECTANGLE rec2

X-Coordinate: 3

Y-Coordinate: 4

Side a: 11

Side b: 7

Area: 77

Perimeter: 36

Expected to display the following values for objec rec2:

Rectangle Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

Side a: 12

Side b: 15

Area: 180

Perimeter: 54

If it doesn't there is a problem with your assignment operator.

Rectangle Name: RECTANGLE A

X-Coordinate: 5

Y-Coordinate: 7

Side a: 12

Side b: 15

Area: 180

Perimeter: 54

Testing copy constructor in class Rectangle:

Rectangle Name: RECTANGLE A

X-Coordinate: 5

Y-Coordinate: 7

Side a: 100

Side b: 200

Area: 20000

Perimeter: 600

Expected to display the following values for objec rec2:

Rectangle Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

Side a: 100

Side b: 200

Area: 20000

Perimeter: 600

If it doesn't there is a problem with your assignment operator.

Rectangle Name: RECTANGLE A

X-Coordinate: 5

Y-Coordinate: 7

Side a: 100

Side b: 200

Area: 20000

Perimeter: 600