Course: ENSF614 – Fall 2025
Lab #: Lab 2
Instructor: Mahmood Moussavi
Student Name: John Zhou
Submission Date: Sep 24th, 2025

I have been keeping all the files in github. I hope by providing this github link will help you a little bit.
https://github.com/JZ-Zhou-UofC/ENSF-614-assignment-repo

Exercise A

```
/*
*
*
File Name: Lab2exAmain.cpp
Assignment: Lab 2 Exercise A
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#include <assert.h>
#include <iostream>
#include "dictionaryList.h"

using namespace std;

DictionaryList dictionary_tests();

void print(DictionaryList& dl);

void test_operator_overloading(DictionaryList& dl);

int main()
{
  DictionaryList dl = dictionary_tests();
#if 1
 test_operator_overloading(dl);
#endif
  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 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';
  }
}

#if 1
void test_operator_overloading(DictionaryList& dl)
{

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

```
dl2.go_to_first();
// Simce datum is a string object from C++ Libray and most of the operators are
//overloaded by class string, operators suc as <. >. >=, << works automatically.

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

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();

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 << "\n Using square bracket [] to access elements of datum 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
```

```
/*
*
*
File Name: dictonaryList.cpp
Assignment: Lab 2 Exercise A
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/

#include <assert.h>
#include <iostream>
#include <stdlib.h>
#include "dictionaryList.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 string& 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;
}

void DictionaryList::find(const Key& keyA)
{
   for (Node *p = headM; p != 0; p=p->nextM)
      if (keyA == p->keyM)
      {
         cout << "'" << keyA <<"' was found with datum value " << p->datumM.c_str() << ".\n";
         cursorM = p;
         return;
      }
   cout << "'" << keyA <<"' was not found.\n";
```

```
    cursorM = 0;
}

void DictionaryList::destroy()
{

        Node *p = headM;
        Node *prev;
        while (p != 0)
        {
            prev = p;
            p = p->nextM;
            delete prev;
        }
        headM = 0;
        sizeM = 0;

}

void DictionaryList::copy(const DictionaryList& source)
{
    if (source.headM == 0) {
        headM = 0;
        return;
    }

    headM = new Node (source.headM->keyM, source.headM->datumM, NULL);
    Node *newest_node = headM;
```

```
    const Node *source_node = source.headM;

    if(source_node == source.cursorM)
        cursorM = newest_node;

    while (true) {
        source_node = source_node->nextM;

        if (source_node == 0)
            break;

        newest_node->nextM = new Node(source_node->keyM, source_node->datumM, NULL);

       if(source_node == source.cursorM)
            cursorM = newest_node->nextM;

        newest_node = newest_node->nextM;

    }

    sizeM = source.sizeM;

}

std::ostream& operator<<(std::ostream& os, const DictionaryList& dl) {
    Node* p = dl.headM;
    while (p != nullptr) {
        os << p->keyM << p->datumM << std::endl;
        p = p->nextM;
```

```
    }
    return os;
}


 Datum& DictionaryList::operator[](int index) {
    assert(index >= 0 && index < sizeM);
    Node* p = headM;
    for (int i = 0; i < index; ++i)
        p = p->nextM;
    return p->datumM;
}
```

```
/*
*
*
File Name: dictonaryList.h
Assignment: Lab 2 Exercise A
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#ifndef DICTIONARY_H
#define DICTIONARY_H
#include <iostream>
#include <string>
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.

// Edit these typedefs to change the key or datum types, if necessary.
typedef int Key;
typedef string 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 DictionaryList;
class Node {
public:
  friend class DictionaryList;
  friend std::ostream& operator<<(std::ostream& os, const DictionaryList& dl);
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.

  Datum& operator[](int index);
  friend std::ostream& operator<<(std::ostream& os, const DictionaryList& dl);

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
```

Output

```
john2@John-Laptop /cygdrive/c/Users/john2/OneDrive/Desktop/uofc/614
$ ./exa

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---------------------------***


Testig 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 datum 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:
8001Allen
8002Peter
8003Sam
8004PointyHair


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 ----------***


john2@John-Laptop /cygdrive/c/Users/john2/OneDrive/Desktop/uofc/614
$
```

Exercise B

```
/*
*
*
File Name: graphicsWorld.cpp
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#include "graphicsWorld.h"
#include "point.h"
#include "square.h"
#include "rectangle.h"
#include <iostream>

using namespace std;

void GraphicsWorld::run() {
    cout << "-----------------------------------------------------\n";
    cout << "Author: John Zhou\n";
    cout << "-----------------------------------------------------\n\n";

#if 1 // Change 0 to 1 to test Point
    cout << "\nTesting class Point:" << endl;

    Point m(6, 8);
    Point n(6, 8);
    n.setx(9);
```

```
    cout << "\nExpected to display 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) << "\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 rectangle a and b is: " << d << endl;

    Rectangle rec1 = a; // copy constructor
```

```
rec1.display();

cout << "\nTesting assignment operator in class Rectangle:" << endl;
Rectangle rec2(3, 4, 11, 7, "RECTANGLE rec2");
rec2.display();

rec2 = a; // assignment operator
a.set_side_b(200);
a.set_side_a(100);

cout << "\nExpected to display the following values for object 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 object rec3:" << 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 copy constructor.\n" << endl;
    rec3.display();
#endif // end of block to test Rectangle

#if 1 // 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;

    for (int i = 0; i < 4; ++i) {
            cout << "\n --------------------------------" << endl;
       sh[i]->display();
    }
     cout << "\nendddddddddddddddddddd" << endl;
#endif // end of block to test array of pointer and polymorphism
}
```

```
 /*
*
*
File Name: graphicsWorld.h
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#ifndef GRAPHICSWORLD_H
#define GRAPHICSWORLD_H

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

#endif
```

```
/*
*
*
File Name: Lab2_exBmain.cpp
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#include "graphicsWorld.h"

int main() {
   GraphicsWorld gw;
   gw.run();
   return 0;
}
```

```
/*
*
*
File Name: point.cpp
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#include "point.h"
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

int Point::idCounter = 1000;
// Constructor
Point::Point(double x_val, double y_val) : x(x_val), y(y_val) {
   idCounter ++;
   id=idCounter;
}

// Getters
double Point::getx() const { return x; }
double Point::gety() const { return y; }
int Point::getId() const { return id; }

// Setters
void Point::setx(double x_val) { x = x_val; }
```

```
void Point::sety(double y_val) { y = y_val; }

void Point::display() const {
    cout << fixed << setprecision(2);
    cout << "X-coordinate: " << x << "\n";
    cout << "Y-coordinate: " << y << "\n";
}

int Point::counter() {
    return idCounter -1000;
}

double Point::distance(const Point& other) const {
    double dx = x - other.x;
    double dy = y - other.y;
    return sqrt(dx * dx + dy * dy);
}

// Static distance (between any two points)
double Point::distance(const Point& p1, const Point& p2) {
    double dx = p1.x - p2.x;
    double dy = p1.y - p2.y;
    return sqrt(dx * dx + dy * dy);
}
```

```
/*
*
*
File Name: point.h
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#ifndef POINT_H
#define POINT_H

class Point {
private:
   double x;
   double y;
   int id;
   static int idCounter; // Static member to track all ID

public:
   Point(double x_val, double y_val);

   // Getters
   double getx() const;
   double gety() const;
   int getId() const;

   // Setters
   void setx(double x_val);
   void sety(double y_val);
```

```
    void display() const;
    //static to access the static member
    static int counter();
    double distance(const Point& other) const;
    //This can be static because this does not require an object of point
    static double distance(const Point& p1, const Point& p2);
};

#endif
```

```
/*
*
*
File Name: rectangle.cpp
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#include "rectangle.h"
#include <iostream>
#include <iomanip>
using namespace std;

Rectangle::Rectangle(double x, double y, double side_a, double side_b, const char *name)
   : Square(x, y, side_a, name), side_b(side_b) {}

Rectangle::Rectangle(const Rectangle &other)
   : Square(other), side_b(other.side_b) {}

Rectangle &Rectangle::operator=(const Rectangle &other)
{
   if (this != &other)
   {
      Square::operator=(other);
      side_b = other.side_b;
   }
   return *this;
}
```

```
double Rectangle::get_side_b() const
{
   return side_b;
}

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

double Rectangle::area() const
{
   return get_side_a() * side_b;
}

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

void Rectangle::display() const
{
   cout << "Rectangle Name: " << (getName()) << "\n";
   cout << fixed << setprecision(2);
   cout << "X-coordinate: " << getOrigin().getx() << "\n";
   cout << "Y-coordinate: " << getOrigin().gety() << "\n";
   cout << "Side a: " << get_side_a() << "\n";
   cout << "Side b: " << side_b << "\n";
   cout << "Area: " << area() << "\n";
```

```
    cout << "Perimeter: " << perimeter() << "\n";
}
```

```
/*
*
*
File Name: rectangle.cpp
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include "square.h"

class Rectangle : public Square
{
private:
   double side_b;

public:
   Rectangle(double x, double y, double side_a, double side_b, const char *name);

   Rectangle(const Rectangle &other);
   Rectangle &operator=(const Rectangle &other);

   double get_side_b() const;
   void set_side_b(double side);

   double area() const override;
   double perimeter() const override;
```

```
    void display() const override;
};


#endif
```

```
/*
*
*
File Name: shape.cpp
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#include "shape.h"
#include <iostream>
#include <cstring>
#include <cmath>
#include <iomanip>
using namespace std;

// Constructor
Shape::Shape(const Point& pt, const char* name) : origin(pt) {

    shapeName = new char[strlen(name) + 1];
    strcpy(shapeName, name);

}

Shape::Shape(const Shape& other) : origin(other.origin) {
  if (other.shapeName) {
    shapeName = new char[strlen(other.shapeName) + 1];
    strcpy(shapeName, other.shapeName);
  } else {
    shapeName = nullptr;
```

```
    }
}

// Assignment operator
Shape& Shape::operator=(const Shape& other) {
    if (this != &other) {
        origin = other.origin;

        delete[] shapeName;
        if (other.shapeName) {
            shapeName = new char[strlen(other.shapeName) + 1];
            strcpy(shapeName, other.shapeName);
        } else {
            shapeName = nullptr;
        }
    }
    return *this;
}
// Destructor
Shape::~Shape() {
    delete[] shapeName;
}

// Getters
const Point& Shape::getOrigin() const {
    return origin;
}

const char* Shape::getName() const {
```

```
    return shapeName;
}

void Shape::move(double dx, double dy) {
    origin.setx(origin.getx() + dx);
    origin.sety(origin.gety() + dy);
}

void Shape::display() const {
    cout << "Shape Name: " << (shapeName ? shapeName : "Unnamed") << "\n";
    cout << fixed << setprecision(2);
    cout << "X-coordinate: " << origin.getx() << "\n";
    cout << "Y-coordinate: " << origin.gety() << "\n";
}

double Shape::distance(Shape& other) {
    return origin.distance(other.getOrigin());
}

double Shape::distance(Shape& s1, Shape& s2) {
    return Point::distance(s1.getOrigin(), s2.getOrigin());
}
```

```
/*
*
*
File Name: shape.h
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#ifndef SHAPE_H
#define SHAPE_H

#include "point.h"

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

public:
   // shapeName is on heap
   // Constructor
   Shape(const Point& pt, const char* name);
   // Copy constructor
   Shape(const Shape& other);
   // Assignment operator
   Shape& operator=(const Shape& other);

   virtual ~Shape();
```

```
    // Getters
    const Point& getOrigin() const;
    const char* getName() const;

    void move(double dx, double dy);

    virtual void display() const;


    double distance(Shape& other);
    // Static function, does not require shape object
    static double distance(Shape& s1, Shape& s2);
};

#endif
```

```
/*
*
*
File Name: square.cpp
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#include "square.h"
#include <iostream>
#include <iomanip>
using namespace std;

// Constructor
Square::Square(double x, double y, double side, const char *name)
   : Shape(Point(x, y), name), side_a(side) {}


double Square::get_side_a() const
{
   return side_a;
}
void Square::set_side_a(double side)
{
   side_a = side;
}

double Square::area() const
{
```

```
    return side_a * side_a;
}

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

void Square::display() const
{
    cout << "Square Name: " << (getName()) << "\n";
    cout << fixed << setprecision(2);
    cout << "X-coordinate: " << getOrigin().getx() << "\n";
    cout << "Y-coordinate: " << getOrigin().gety() << "\n";
    cout << "Side a: " << side_a << "\n";
    cout << "Area: " << area() << "\n";
    cout << "Perimeter: " << perimeter() << "\n";
}
```

```
/*
*
*
File Name: square.h
Assignment: Lab 2 Exercise B
*  Completed by: John Zhou
*  Submission Date: Sept 24, 2025
*/
#ifndef SQUARE_H
#define SQUARE_H

#include "shape.h"

class Square : public Shape
{
private:
   double side_a;

public:
   // Constructor
   Square(double x, double y, double side, const char *name);

   double get_side_a() const;
   void set_side_a(double side);

  virtual double area() const ;
   virtual double perimeter() const ;
   virtual void display() const ;
};
```

```
#endif
```

```
john2@John-Laptop /cygdrive/c/Users/john2/OneDrive/Desktop/uofc/614/ExerciseB
$ ./exB.exe
--------------------------------------------------
Author: John Zhou
--------------------------------------------------


Testing class Point:

Expected to display 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: 5.00
Y-coordinate: 7.00
Side a: 12.00
Area: 144.00
Perimeter: 48.00

Testing Functions in class Rectangle:
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Side b: 15.00
Area: 180.00
Perimeter: 54.00
Rectangle Name: RECTANGLE B
X-coordinate: 16.00
Y-coordinate: 7.00
Side a: 8.00
Side b: 9.00
Area: 72.00
Perimeter: 34.00

Distance between rectangle a and b is: 11.00
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Side b: 15.00
Area: 180.00
Perimeter: 54.00

Testing assignment operator in class Rectangle:
Rectangle Name: RECTANGLE rec2
X-coordinate: 3.00
Y-coordinate: 4.00
Side a: 11.00
Side b: 7.00
Area: 77.00
Perimeter: 36.00

Expected to display the following values for object rec2:
Rectangle Name: RECTANGLE A
X-coordinate: 5
```

```
Side b: 200.00
Area: 20000.00
Perimeter: 600.00

Expected to display the following values for object rec3:
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 copy constructor.

Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 100.00
Side b: 200.00
Area: 20000.00
Perimeter: 600.00

Testing array of pointers and polymorphism:

 --------------------------------
Square Name: SQUARE - S
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Area: 144.00
Perimeter: 48.00

 --------------------------------
Rectangle Name: RECTANGLE B
X-coordinate: 16.00
Y-coordinate: 7.00
Side a: 8.00
Side b: 9.00
Area: 72.00
Perimeter: 34.00

 --------------------------------
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 12.00
Side b: 15.00
Area: 180.00
Perimeter: 54.00

 --------------------------------
Rectangle Name: RECTANGLE A
X-coordinate: 5.00
Y-coordinate: 7.00
Side a: 100.00
Side b: 200.00
Area: 20000.00
Perimeter: 600.00

enddddddddddddddddddd

john2@John-Laptop /cygdrive/c/Users/john2/OneDrive/Desktop/uofc/614/ExerciseB
$
```