Course: Programming Fundamental - ENSF 480
Lab #: Lab 3
Instructor: G. Gouri
Student Name: Daniel Rey, Aly Farouz
Lab Section: B01
Date Submitted: September 26, 2025

Exercise A

```
/*
* File Name: circle.h
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

#ifndef CIRCLE_H
#define CIRCLE_H
#include "shape.h"
using namespace std;

class Circle: virtual public Shape{
protected:
  double radius;
public:
  Circle(double x, double y, double r, const char* name);
  Circle(const Circle& other);
  Circle& operator =(const Circle& rhs);

  const double getRadius()const;
  virtual void setRadius(double r);

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

```
/*
* File Name: curveCut.h
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

#ifndef CURVECUT_H
#define CURVECUT_H
#include "circle.h"
#include "rectangle.h"
using namespace std;

class CurveCut: public Rectangle, public Circle{
public:
  CurveCut(double x, double y, double a, double b, double r, const char* name);
  CurveCut(const CurveCut& other);
  CurveCut& operator =(const CurveCut& rhs);

  void setRadius(double r);
  void set_side_b(double b);
  void set_side_a(double a);

  double area()const;
  double perimeter()const;
  void display()const;
};
#endif
```

```
/*
* File Name: graphicsWorld.h
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

#ifndef GRAPHICSWORLD_H
#define GRAPHICSWORLD_H
#include "point.h"
#include "shape.h"
#include "square.h"
#include "rectangle.h"
using namespace std;

class GraphicsWorld {
public:
  GraphicsWorld();
  ~GraphicsWorld();
  static void run();
};
#endif
```

```
/*
* File Name: point.h
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

#ifndef POINT_H
#define POINT_H
using namespace std;

class Point{
private:
  double x;
  double y;
  static int count;
  const int id;
public:
  Point(double x, double y);
  Point(const Point& other);
  Point& operator=(const Point& rhs);
  double getx() const;
  void setx(double a);
  double gety() const;
  void sety(double a);

  void display()const;
  const static int counter();
  static double distance(const Point& the_point, const Point& other);
  double distance(const Point& other)const;
};
#endif
```

```
/*
* File Name: rectangle.h
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/
```

```
#ifndef RECTANGLE_H
#define RECTANGLE_H
#include "square.h"
using namespace std;

class Rectangle: virtual public Square{
protected:
  double side_b;
public:
  Rectangle(double x, double y, double a, double b, const char* name);
  Rectangle(const Rectangle& other);
  Rectangle& operator =(const Rectangle& rhs);
  const double get_side_b()const;
  virtual void set_side_b(double b);

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

```
/*
* File Name: shape.h
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

#ifndef SHAPE_H
#define SHAPE_H
#include "point.h"
using namespace std;

class Shape{
protected:
  Point origin;
  char* shapeName;
public:
  Shape(double x, double y, const char* name);
  Shape(const Shape& other);
  virtual ~Shape();
  Shape& operator=(const Shape& rhs);
  const Point& getOrigin() const;
  const char* getName() const;

  virtual void display()const;
  double distance(const Shape& other)const;
  static double distance(const Shape& the_shape, const Shape& other);
  void move(double dx, double dy);
  virtual double area()const;
  virtual double perimeter()const;
};
#endif
```

```
/*
* File Name: square.h
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

#ifndef SQUARE_H
#define SQUARE_H
#include "shape.h"
using namespace std;
```

```
class Square: virtual public Shape{
protected:
  double side_a;
public:
  Square(double x, double y, double a, const char* name);
  Square(const Square& other);
  Square& operator =(const Square& rhs);
  const double get_side_a()const;
  void set_side_a(double a);

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

```
/*
* File Name: circle.cpp
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

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

Circle::Circle(double x, double y, double r, const char* name):
Shape(x,y,name),
radius(r){}

Circle::Circle(const Circle& other):
Shape(other),
radius(other.getRadius()){}

Circle& Circle::operator=(const Circle& rhs){
  if (this!=&rhs){
    Shape::operator=(rhs);
    radius = rhs.getRadius();
  }
  return *this;
}

const double Circle::getRadius()const{return radius;}
void Circle::setRadius(double r){radius=r;}

double Circle::area()const{return radius*radius*M_PI;}
double Circle::perimeter()const{return 2*radius*M_PI;}

void Circle::display()const{
  cout << "Circle Name: " << shapeName << endl;
  origin.display();
  cout << "Radius: " << radius << endl
       << "Area: " << area() << endl
       << "Perimeter: " << perimeter() << endl;
}
```

```
/*
* File Name: curveCut.cpp
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/
```

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

CurveCut::CurveCut(double x, double y, double a, double b, double r, const char* name):
Shape(x,y,name),
Square(x,y,a,name),
Rectangle(x,y,a,b,name),
Circle(x,y,r,name){
  if (r>a||r>b) {
    cout << "Radius must not be greater than either of the rectangle's sides" << endl;
    exit(1);
  }
}

CurveCut::CurveCut(const CurveCut& other):
Shape(other),
Square(other),
Rectangle(other),
Circle(other){}

CurveCut& CurveCut::operator=(const CurveCut& rhs){
  if (this!=&rhs){
    Rectangle::operator=(rhs);
    radius = rhs.getRadius();
  }
  return *this;
}

void CurveCut::setRadius(double r){
  if (r>side_a||r>side_b) {
    cout << "Radius must not be greater than either of the rectangle's sides" << endl;
    exit(1);
  }
  radius=r;
}
void CurveCut::set_side_b(double b){
  if (radius>b) {
    cout << "Radius must not be greater than either of the rectangle's sides" << endl;
    exit(1);
  }
  side_b=b;
}
void CurveCut::set_side_a(double a){
  if (radius>a) {
    cout << "Radius must not be greater than either of the rectangle's sides" << endl;
    exit(1);
  }
  side_a=a;
}

double CurveCut::area()const{return Rectangle::area()-Circle::area()/4;}
double CurveCut::perimeter()const{return
Rectangle::perimeter()-radius*2+Circle::perimeter()/4;}

void CurveCut::display()const{
  cout << "CurveCut Name: " << shapeName << endl;
  origin.display();
  cout << "Width: " << side_a << endl
       << "Length: " << side_b << endl
       << "Radius of the cut: " << radius << endl;
}
```

```
/*
* File Name: graphicsWorld.cpp
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

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

void GraphicsWorld::run(){
  #if 0 // Change 0 to 1 to test Point
    Point m (6, 8);
    Point n (6,8);
    n.setx(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 copy constructor.\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
        // Note that Recatngle and Square tests must alo be run
    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
  #if 1 // Change 0 to 1 to test Circle and CurveCut
        // Note that Rectangle and Square tests must also be run
    cout << "\nTesting Functions in class Circle:" <<endl;
    Circle c (3, 5, 9, "CIRCLE C");
    c.display();
    cout << "the area of " << c.getName() <<" is: "<< c.area() << endl;
    cout << "the perimeter of " << c.getName() << " is: "<< c.perimeter() << endl;
    d = a.distance(c);
    cout << "\nThe distance between rectangle a and circle c is: " << d << endl;

    CurveCut rc (6, 5, 10, 12, 9, "CurveCut rc");
    rc.display();
    cout << "the area of " << rc.getName() <<" is: "<< rc.area() << endl;
    cout << "the perimeter of " << rc.getName() << " is: "<< rc.perimeter();
    d = rc.distance(c);
    cout << "\nThe distance between rc and c is: " << d << endl;


    // Using array of Shape pointers:
    Shape* sh[4];
    sh[0] = &s;
    sh[1] = &a;
    sh [2] = &c;
    sh [3] = &rc;
    sh [0]->display();
    cout << "\nthe area of "<< sh[0]->getName() << " is: "<< sh[0] ->area();
    cout << "\nthe perimeter of " << sh[0]->getName () << " is: "<< sh[0]->perimeter();
    sh [1]->display();
    cout << "\nthe area of "<< sh[1]->getName() << " is: "<< sh[1] ->area();
    cout << "\nthe perimeter of " << sh[0]->getName () << " is: "<< sh[1]->perimeter();
    sh [2]->display();
    cout << "\nthe area of "<< sh[2]->getName() << " is: "<< sh[2] ->area();
    cout << "\nthe circumference of " << sh[2]->getName ()<< " is: "<< sh[2]->perimeter();
    sh [3]->display();
    cout << "\nthe area of "<< sh[3]->getName() << " is: "<< sh[3] ->area();
    cout << "\nthe perimeter of " << sh[3]->getName () << " is: "<< sh[3]->perimeter();

    cout << "\n\nTesting copy constructor in class CurveCut:" <<endl;
    CurveCut cc = rc;
    cc.display();

    cout << "\nTesting assignment operator in class CurveCut:" <<endl;
    CurveCut cc2(2, 5, 100, 12, 9, "CurveCut cc2");
    cc2.display();
    cc2 = cc;
    cc2.display();
  #endif
} // END OF FUNCTION run

#if 1
main(){GraphicsWorld::run();}
// g++ -Wall graphicsWorld.cpp point.cpp shape.cpp square.cpp rectangle.cpp circle.cpp
curveCut.cpp -o exA.exe
#endif
```

```
/*
* File Name: point.cpp
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

#ifndef POINT_H
#include "point.h"
int Point::count = 0;
#endif


Point::Point(double h, double v): x(h), y(v), id(++count+1000){}

Point::Point(const Point& other): x(other.getx()), y(other.gety()), id(++count+1000){}

Point& Point::operator=(const Point& rhs){
  x=rhs.getx();
  y=rhs.gety();
  return *this;
}

double Point::getx()const{return x;}
void Point::setx(double a){x=a;}
double Point::gety()const{return y;}
void Point::sety(double a){y=a;}

void Point::display()const{
  cout << "X-coordinate: ";
  cout << right << setw(9) << fixed << setprecision(2) << this->x << endl;
  cout << "Y-coordinate: ";
  cout << right << setw(9) << fixed << setprecision(2) << this->y << endl;
}

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

double Point::distance(const Point& the_point, const Point& other){
  double dx=the_point.getx()-other.getx();
  double dy=the_point.gety()-other.gety();
  return sqrt(dx*dx+dy*dy);
}
double Point::distance(const Point& other)const{
  double dx=this->x-other.getx();
  double dy=this->y-other.gety();
  return sqrt(dx*dx+dy*dy);
}



/*
* File Name: rectangle.cpp
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

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

```
Rectangle::Rectangle(double x, double y, double a, double b, const char* name):
Shape(x,y,name),
Square(x,y,a,name),
side_b(b){}

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

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

const double Rectangle::get_side_b()const{return side_b;}
void Rectangle::set_side_b(double b){side_b=b;}

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

void Rectangle::display()const{
  cout << "Rectangle Name: " << shapeName << endl;
  origin.display();
  cout << "Side a: " << side_a << endl
       << "Side b: " << side_b << endl
       << "Area: " << area() << endl
       << "Perimeter: " << perimeter() << endl;
}



/*
* File Name: shape.cpp
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

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

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

Shape::Shape(const Shape& other):
origin(Point(other.getOrigin())),
shapeName(new char[strlen(other.getName())+1]){
  strcpy(this->shapeName, other.getName());
}

Shape::~Shape(){
  delete [] shapeName;
}
```

```
Shape& Shape::operator=(const Shape& rhs){
  if (this!=&rhs){
    delete [] shapeName;
    shapeName = new char[strlen(rhs.getName())+1];
    strcpy(this->shapeName, rhs.getName());
    origin = rhs.getOrigin();
  }
  return *this;
}

const Point& Shape::getOrigin()const{
  const Point& p=origin;
  return p;
}

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


void Shape::display()const{
  cout << "Shape Name: " << shapeName << endl;
  origin.display();
}

double Shape::distance(const Shape& other)const{
  return Point::distance(this->origin, other.getOrigin());
}
double Shape::distance(const Shape& the_shape, const Shape& other){
  return Point::distance(the_shape.getOrigin(), other.getOrigin());
}

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

double Shape::area()const{return 0;}
double Shape::perimeter()const{return 0;}



/*
* File Name: square.cpp
* Assignment: Lab 3, Exercise A
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/

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

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

Square::Square(const Square& other): Shape(other), side_a(other.get_side_a()){}

Square& Square::operator=(const Square& rhs){
  if (this!=&rhs){
    Shape::operator=(rhs);
    side_a = rhs.get_side_a();
  }
  return *this;
}

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

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

```
void Square::display()const{
  cout << "Square Name: " << shapeName << endl;
  origin.display();
  cout << "Side a: " << side_a << endl
       << "Area: " << area() << endl
       << "Perimeter: " << perimeter() << endl;
}
```

```
HPLaptop@DESKTOP-C6NJ9VH ~/ENSF480/lab3/exA
$ g++ -Wall graphicsWorld.cpp point.cpp shape.cpp square.cpp rectangle.cpp circl
e.cpp curveCut.cpp -o exA.exe

HPLaptop@DESKTOP-C6NJ9VH ~/ENSF480/lab3/exA
$$ ./exA.exe


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 square 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 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:      5.00
Y-coordinate:      7.00
Side a: 12.00
Side b: 15.00
Area: 180.00
Perimeter: 54.00

Testing copy constructor in class Rectangle:
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

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 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 Functions in class Circle:
Circle Name: CIRCLE C
X-coordinate:      3.00
Y-coordinate:      5.00
Radius: 9.00
Area: 254.47
Perimeter: 56.55
the area of CIRCLE C is: 254.47
the perimeter of CIRCLE C is: 56.55

The distance between rectangle a and circle c is: 2.83
```

```
CurveCut Name: CurveCut rc
X-coordinate:      6.00
Y-coordinate:      5.00
Width: 10.00
Length: 12.00
Radius of the cut: 9.00
the area of CurveCut rc is: 56.38
the perimeter of CurveCut rc is: 40.14
The distance between rc and c is: 3.00
Square Name: SQUARE - S
X-coordinate:      5.00
Y-coordinate:      7.00
Side a: 12.00
Area: 144.00
Perimeter: 48.00

the area of SQUARE - S is: 144.00
the perimeter of SQUARE - S is: 48.00
Rectangle Name: RECTANGLE A
X-coordinate:      5.00
Y-coordinate:      7.00
Side a: 400.00
Side b: 300.00
Area: 120000.00
Perimeter: 1400.00

the area of RECTANGLE A is: 120000.00
the perimeter of SQUARE - S is: 1400.00
Circle Name: CIRCLE C
X-coordinate:      3.00
Y-coordinate:      5.00
Radius: 9.00
Area: 254.47
Perimeter: 56.55

the area of CIRCLE C is: 254.47
the circumference of CIRCLE C is: 56.55
CurveCut Name: CurveCut rc
X-coordinate:      6.00
Y-coordinate:      5.00
Width: 10.00
Length: 12.00
Radius of the cut: 9.00

the area of CurveCut rc is: 56.38
the perimeter of CurveCut rc is: 40.14

Testing copy constructor in class CurveCut:
CurveCut Name: CurveCut rc
X-coordinate:      6.00
Y-coordinate:      5.00
Width: 10.00
Length: 12.00
Radius of the cut: 9.00

Testing assignment operator in class CurveCut:
```

```
CurveCut Name: CurveCut cc2
X-coordinate:      2.00
Y-coordinate:      5.00
Width: 100.00
Length: 12.00
Radius of the cut: 9.00
CurveCut Name: CurveCut rc
X-coordinate:      6.00
Y-coordinate:      5.00
Width: 10.00
Length: 12.00
Radius of the cut: 9.00
```

Exercise B

```
/*
* File Name: mystring2.h
* Assignment: Lab 3, Exercise B
* Lab Section: B01
* Modified by: Daniel Rey
* Submission Date: Sept 26, 2025
*/


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

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.

  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 operator>( const Mystring& s)const;
  // REQUIRES: s refers to an object of class Mystring
  // PROMISES: retruns true if charsM is greater than s.charsM.

  int operator<(const Mystring& s)const;
  // REQUIRES: s refers to an object of class Mystring
  // PROMISES: retruns true if charsM is less than s.charsM.

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

  int operator!=(const Mystring& s)const;
  // REQUIRES: s refers to an object of class Mystring
  // PROMISES: retruns true if charsM is not equal s.charsM.

  friend ostream& operator<<(ostream& os, const Mystring& s);

 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: mystring2.h
* Assignment: Lab 3, Exercise B
* Lab Section: B01
* Modified by: Daniel Rey
* Submission Date: Sept 26, 2025
*/


#include "mystring2.h"
#include <string.h>
#include <iostream>
using namespace std;

Mystring::Mystring()
{
  charsM = new char[1];
  charsM[0] = '\0';
  lengthM = 0;
}
```

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

Mystring::Mystring(int n)
  : lengthM(0), charsM(new char[n])
{
  charsM[0] = '\0';
}

Mystring::Mystring(const Mystring& source):
  lengthM(source.lengthM), charsM(new char[source.lengthM+1])
{
  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 = strlen(S.charsM);
  charsM = new char [lengthM+1];
  strcpy(charsM,S.charsM);
  return *this;
}
```

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

    strcpy(charsM, s);
}

//functions edited by student below this point

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

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

int Mystring::operator>(const Mystring& s)const
{
  return (strcmp(charsM, s.charsM)> 0);
}

int Mystring::operator<(const Mystring& s)const
{
  return (strcmp(charsM, s.charsM)< 0);
}

ostream& operator<<(ostream& os, const Mystring& s){
  return os << s.c_str();
}
```

```
/*
* File Name: iterator.cpp
* Assignment: Lab 3, Exercise B
* Lab Section: B01
* Completed by: Daniel Rey
* Submission Date: Sept 26, 2025
*/


#include <iostream>
#include <assert.h>
#include "mystring2.h"
using namespace std;

template <class T = int>
class Vector {
public:
```

```
  class VectIter{//FIXME this probably needs a template or something
    friend class Vector;
  private:
    Vector *v; // points to a vector object of type T
    int index;    // represents the subscript number of the vector's
                  // array.
  public:
    VectIter(Vector& x);

    T operator++();
    //PROMISES: increments the iterator's indes and return the
    //          value of the element at the index position. If
    //          index exceeds the size of the array it will
    //          be set to zero. Which means it will be circulated
    //          back to the first element of the vector.

    T  operator++(int);
    // PRIMISES: returns the value of the element at the index
    //           position, then increments the index. If
    //           index exceeds the size of the array it will
    //           be set to zero. Which means it will be circulated
    //           back to the first element of the vector.

    T  operator--();
    // PROMISES: decrements the iterator index, and return the
    //           the value of the element at the index. If
    //           index is less than zero it will be set to the
    //           last element in the aray. Which means it will be
    //           circulated to the last element of the vector.

    T  operator--(int);
    // PRIMISES: returns the value of the element at the index
    //           position, then decrements the index. If
    //           index is less than zero it will be set to the
    //           last element in the aray. Which means it will be
    //           circulated to the last element of the vector.

    T operator *();
    // PRIMISES: returns the value of the element at the current
    //           index position.
  };

  Vector(int sz);
  ~Vector();

  T & operator[](int i);
  // PRIMISES: returns existing value in the ith element of
  //           array or sets a new value to  the ith element in
  //           array.

  void ascending_sort();
  // PRIMISES: sorts the vector values in ascending order.

private:
  T *array;               // points to the first element of an array of T
  int size;               // size of array
  void swap(T&, T&); // swaps the values of two elements in array
};


template <class T>
void Vector<T>::ascending_sort()
{
  for(int i=0; i< size-1; i++)
    for(int j=i+1; j < size; j++)
      if(array[i] > array[j])
        swap(array[i], array[j]);
}
```

```
template <>
void Vector<const char*>::ascending_sort()
{
  for(int i=0; i< size-1; i++)
    for(int j=i+1; j < size; j++)
      for(int k=0; 1; k++)
        if (array[i][k]!=array[j][k] || array[i][k]=='\0' || array[j][k]=='\0'){
          if (array[i][k] > array[j][k]){
            swap(array[i], array[j]);
          }
          break;
        }
}

template <class T>
void Vector<T>::swap(T& a, T& b)
{
  T tmp = a;
  a = b;
  b = tmp;
}

template <class T>
T Vector<T>::VectIter::operator++()
{
  if (++index==v->size){
    index = 0;
  }
  return v->array[index];
}

template <class T>
T Vector<T>::VectIter::operator++(int)
{
  if (index==v->size-1){
    index = 0;
    return v->array[v->size-1];
  } else {
    return v->array[index++];
  }
}

template <class T>
T Vector<T>::VectIter::operator--()
{
  if (index==0){
    index = v->size;
  }
  return v->array[--index];
}

template <class T>
T Vector<T>::VectIter::operator--(int)
{
  if (index==0){
    index = v->size-1;
    return v->array[0];
  } else {
    return v->array[index--];
  }
}

template <class T>
T Vector<T>::VectIter::operator *()
{
  return v -> array[index];
}
```

```
template <class T>
Vector<T>::VectIter::VectIter(Vector& x)
{
  v = &x;
  index = 0;
}

template <class T>
Vector<T>::Vector(int sz)
{
  size=sz;
  array = new T [sz];
  assert (array != NULL);
}

template <class T>
Vector<T>::~Vector()
{
  delete [] array;
  array = NULL;
}

template <class T>
T& Vector<T>::operator [] (int i)
{
  return array[i];
}


int main()
{

#if 0
 Vector x(3);
 x[0] = 999;
 x[1] = -77;
 x[2] = 88;

 Vector::VectIter iter(x);

 cout << "\n\nThe first element of vector x contains: " << *iter;

 // the code between the  #if 0 and #endif is ignored by
 // compiler. If you change it to #if 1, it will be compiled

  cout << "\nTesting an <int> Vector: " << endl;

  cout << "\n\nTesting sort";
  x.ascending_sort();

  for (int i=0; i<3 ; i++)
    cout << endl << iter++;

  cout << "\n\nTesting Prefix --:";
  for (int i=0; i<3 ; i++)
    cout << endl << --iter;

  cout << "\n\nTesting Prefix ++:";
  for (int i=0; i<3 ; i++)
    cout << endl << ++iter;

  cout << "\n\nTesting Postfix --";
  for (int i=0; i<3 ; i++)
    cout << endl << iter--;

  cout << endl;
#endif
#if 1
  cout << "Testing a <Mystring> Vector: " << endl;
  Vector<Mystring> y(3);
  y[0] = "Bar";
```

```
  y[1] = "Foo";
  y[2] = "All";;

  Vector<Mystring>::VectIter iters(y);

  cout << "\n\nTesting sort";
  y.ascending_sort();

  for (int i=0; i<3 ; i++)
    cout << endl << iters++;

  cout << "\n\nTesting Prefix --:";
  for (int i=0; i<3 ; i++)
    cout << endl << --iters;

  cout << "\n\nTesting Prefix ++:";
  for (int i=0; i<3 ; i++)
    cout << endl << ++iters;

  cout << "\n\nTesting Postfix --";
  for (int i=0; i<3 ; i++)
    cout << endl << iters--;

  cout << endl; cout << "Testing a <char *> Vector: " << endl;
  Vector<const char*> z(3);
  z[0] = "Orange";
  z[1] = "Pear";
  z[2] = "Apple";;

  Vector<const char*>::VectIter iterchar(z);

  cout << "\n\nTesting sort";//FIXME does not work for this type
  z.ascending_sort();

  for (int i=0; i<3 ; i++)
    cout << endl << iterchar++;

#endif
  cout << "\nProgram Terminated Successfully." << endl;

  return 0;
}
```

```
HPLaptop@DESKTOP-C6NJ9VH ~/ENSF480/lab2/exB
$ g++ -Wall iterator.cpp mystring2.cpp -o exB.exe

HPLaptop@DESKTOP-C6NJ9VH ~/ENSF480/lab2/exB
$$ ./exB


The first element of vector x contains: 999
Testing an <int> Vector:


Testing sort
-77
88
999
```

```
Testing Prefix --:
999
88
-77

Testing Prefix ++:
88
999
-77

Testing Postfix --
-77
999
88
Testing a <Mystring> Vector:


Testing sort
All
Bar
Foo

Testing Prefix --:
Foo
Bar
All

Testing Prefix ++:
Bar
Foo
All

Testing Postfix --
All
Foo
Bar
Testing a <char *> Vector:


Testing sort
Apple
Orange
Pear
Program Terminated Successfully.
```

Exercise C

```
/*
* File Name: LookupTable.h
* Assignment: Lab 3, Exercise C
* Lab Section: B01
* Completed by: Aly Farouz
* Submission Date: Sept 26, 2025
*/



#ifndef LOOKUPTABLE_H
#define LOOKUPTABLE_H
#include <iostream>
using namespace std;
```

```
// class LookupTable: 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 LookupTable has an embedded iterator class that allows users
//    of the class to traverse trhough the list and  have acess to each
//    node.



#include "customer.h"



// template <typename keyType, typename datumType>
//    In this version of the LookupTable a new struct type called Pair
//    is introduced which represents a key/data pair.









// typedef int LT_Key;
// typedef Customer LT_Datum;



template <typename keyType, typename datumType>
struct Pair
{
 keyType key;
 datumType datum;
 Pair(const keyType &key1, const datumType &datum1)
     : key(key1), datum(datum1) {}
};



template <typename keyType, typename datumType>
class LT_Node {
//  friend class LookupTable;
public:
Pair<keyType, datumType> pairM;
LT_Node<keyType, datumType> *nextM;



// This ctor should be convenient in insert and copy operations.
LT_Node(const Pair<keyType, datumType>& pairA, LT_Node<keyType, datumType> *nextA)
   : pairM(pairA), nextM(nextA) {}
};
```

```
template <typename keyType, typename datumType>
class LookupTable {
public:



// Nested class
class Iterator {
  // friend class LookupTable ;
  LookupTable<keyType, datumType> *LT;
//    LT_Node* cursor;
 public:
  Iterator():LT(0){}
  Iterator(LookupTable & x): LT(&x){}
  const datumType&  operator *();
  const datumType& operator ++();
  const datumType& operator ++(int);
  int operator !();



  void step_fwd(){  assert(LT->cursor_ok());
  LT->step_fwd();}
};



LookupTable();
LookupTable(const LookupTable  & source);
LookupTable& operator =(const LookupTable& rhs);
~LookupTable();



LookupTable<keyType, datumType>& begin();



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



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

```
void insert(const Pair<keyType, datumType>& pariA);
// 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 keyType& 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 keyType& 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.







private:
int sizeM;
LT_Node<keyType, datumType> *headM;
LT_Node<keyType, datumType> *cursorM;



void destroy();
// Deallocate all nodes, set headM to zero.
 void copy(const LookupTable& 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
template <typename keyType, typename datumType>
LookupTable<keyType, datumType>& LookupTable<keyType, datumType>::begin(){
cursorM = headM;
return *this;
}
// template <typename keyType, typename datumType>
// LT_Node<keyType, datumType>::LT_Node(const Pair<keyType, datumType>& pairA, LT_Node *nextA)
//  : pairM(pairA), nextM(nextA)
// {
// }
template <typename keyType, typename datumType>
LookupTable<keyType, datumType>::LookupTable()
: sizeM(0), headM(0), cursorM(0)
{
}
template <typename keyType, typename datumType>
LookupTable<keyType, datumType>::LookupTable(const LookupTable<keyType, datumType>& source)
{
copy(source);
}
template <typename keyType, typename datumType>
LookupTable<keyType, datumType>& LookupTable<keyType, datumType>::operator =(const
LookupTable<keyType, datumType>& rhs)
{
if (this != &rhs) {
  destroy();
  copy(rhs);
}
return *this;
}
template <typename keyType, typename datumType>
LookupTable<keyType, datumType>::~LookupTable()
{
destroy();
}
template <typename keyType, typename datumType>
int LookupTable<keyType, datumType>::size() const
{
return sizeM;
}
template <typename keyType, typename datumType>
int LookupTable<keyType, datumType>::cursor_ok() const
{
return cursorM != 0;
}
template <typename keyType, typename datumType>
const keyType& LookupTable<keyType, datumType>::cursor_key() const
{
assert(cursor_ok());
return cursorM->pairM.key;
}
template <typename keyType, typename datumType>
const datumType& LookupTable<keyType, datumType>::cursor_datum() const
{
assert(cursor_ok());
return cursorM->pairM.datum;
}
template <typename keyType, typename datumType>
void LookupTable<keyType, datumType>::insert(const Pair<keyType, datumType>& pairA)
{
// Add new node at head?
if (headM == 0 || pairA.key < headM->pairM.key) {
  headM = new LT_Node<keyType, datumType>(pairA, headM);
  sizeM++;
```

```
}



// Overwrite datum at head?
else if (pairA.key == headM->pairM.key)
  headM->pairM.datum = pairA.datum;



// Have to search ...



else {
  LT_Node<keyType, datumType>* before= headM;
  LT_Node<keyType, datumType>* after=headM->nextM;



  while(after!=NULL && (pairA.key > after->pairM.key))
    {
before=after;
after=after->nextM;
    }
   if(after!=NULL && pairA.key == after->pairM.key)
    {
after->pairM.datum = pairA.datum;
    }
  else
    {
before->nextM = new LT_Node<keyType, datumType> (pairA, before->nextM);
sizeM++;
    }
}
}
template <typename keyType, typename datumType>
void LookupTable<keyType, datumType>::remove(const keyType& keyA)
{



if (headM == 0 || keyA < headM->pairM.key)
  return;



LT_Node<keyType, datumType>* doomed_node = 0;
if (keyA == headM->pairM.key) {
  doomed_node = headM;
  headM = headM->nextM;
  sizeM--;
}
else {
  LT_Node<keyType, datumType>      *before = headM;
  LT_Node<keyType, datumType> *maybe_doomed = headM->nextM;
  while(maybe_doomed != 0 && keyA > maybe_doomed->pairM.key) {
    before = maybe_doomed;
    maybe_doomed = maybe_doomed->nextM;
  }



  if (maybe_doomed != 0 && maybe_doomed->pairM.key == keyA) {
```

```
    doomed_node = maybe_doomed;
    before->nextM = maybe_doomed->nextM;
    sizeM--;
  }
}
delete doomed_node;           // Does nothing if doomed_node == 0.
}
template <typename keyType, typename datumType>
void LookupTable<keyType, datumType>::find(const keyType& keyA)
{
LT_Node<keyType, datumType> *ptr=headM;
while (ptr != NULL && ptr->pairM.key != keyA)
  {
   ptr=ptr->nextM;
  }



 cursorM = ptr;
}
template <typename keyType, typename datumType>
void LookupTable<keyType, datumType>::go_to_first()
{
cursorM = headM;
}
template <typename keyType, typename datumType>
void LookupTable<keyType, datumType>::step_fwd()
{
assert(cursor_ok());
cursorM = cursorM->nextM;
}
template <typename keyType, typename datumType>
void LookupTable<keyType, datumType>::make_empty()
{
destroy();
sizeM = 0;
cursorM = 0;
}
template <typename keyType, typename datumType>
void LookupTable<keyType, datumType>::destroy()
{



LT_Node<keyType, datumType> *ptr = headM;
while (ptr!=NULL)
  {
    headM=headM->nextM;
    delete ptr;
    ptr=headM;



  }
cursorM = NULL;
sizeM=0;
}
template <typename keyType, typename datumType>
void LookupTable<keyType, datumType>::copy(const LookupTable<keyType, datumType>& source)
{



headM=0;
cursorM =0;
```

```
if(source.headM ==0)
  return;
for(LT_Node<keyType, datumType> *p = source.headM; p != 0; p=p->nextM)
  {
    insert(Pair<keyType, datumType> (p->pairM.key, p->pairM.datum));
    if(source.cursorM == p)
find(p->pairM.key);
  }




}
template <typename keyType, typename datumType>
ostream& operator <<   (ostream& os, const LookupTable<keyType, datumType>& lt)
{
if (lt.cursor_ok())
  os <<lt.cursor_key() << "  " << lt.cursor_datum();
else
  os<<"Not Found.";




return os;
}
template <typename keyType, typename datumType>
const datumType& LookupTable<keyType, datumType>::Iterator::operator *()
{
assert(LT ->cursor_ok());
return LT->cursor_datum();
}
template <typename keyType, typename datumType>
const datumType& LookupTable<keyType, datumType>::Iterator::operator ++()
{
assert(LT->cursor_ok());
const datumType & x = LT->cursor_datum();
LT->step_fwd();
return x;
}
template <typename keyType, typename datumType>
const datumType& LookupTable<keyType, datumType>::Iterator::operator ++(int)
{
assert(LT->cursor_ok());
LT->step_fwd();
return LT->cursor_datum();
}
template <typename keyType, typename datumType>
int LookupTable<keyType, datumType>::Iterator::operator!()
{
return (LT->cursor_ok());
}
#endif
```

```
/*
* File Name: mainLab3ExC.cpp
* Assignment: Lab 3, Exercise C
* Lab Section: B01
* Completed by: Aly Farouz
* Submission Date: Sept 26, 2025
*/


#include <assert.h>
#include <iostream>
#include "lookupTable.h"
#include "customer.h"
#include <cstring>
using namespace std;


template <typename keyType, typename datumType>
void print(LookupTable<keyType,datumType>& lt);


template <typename keyType, typename datumType>
void try_to_find(LookupTable<keyType,datumType>& lt, int key);


template <typename keyType, typename datumType>
void test_Customer();


template <typename keyType, typename datumType>
void test_String();
template <typename keyType, typename datumType>
void test_integer();


int main()
{


//create and test a lookup table with an integer key value and Customer datum
test_Customer<int, Customer>();

// Uncomment the following function calls when ready to test template class LookupTable
// create and test a a lookup table of type <int, String>
test_String<int, Mystring>();


// Uncomment the following function calls when ready to test template class LookupTable
// create and test a a lookup table of type <int, int>
test_integer<int, int>();


 cout<<"\n\nProgram terminated successfully.\n\n";


 return 0;
}
template <typename keyType, typename datumType>
void print(LookupTable<keyType,datumType>& lt)
{
 if (lt.size() == 0)
```

```
   cout << "  Table is EMPTY.\n";
 for (lt.go_to_first(); lt.cursor_ok(); lt.step_fwd()) {
   cout << lt << endl;
 }
}
template <typename keyType, typename datumType>
void try_to_find(LookupTable<keyType,datumType>& lt, int key)
{
 lt.find(key);
 if (lt.cursor_ok())
   cout << "\nFound key:"  << lt;
 else
   cout << "\nSorry, I couldn't find key: " << key << " in the table.\n";
}
template <typename keyType, typename datumType>
void test_Customer()
 //creating a lookup table for customer objects.
 {
   cout<<"\nCreating and testing Customers Lookup Table <not template>-...\n";
   LookupTable<keyType,datumType> lt;

   // Insert using new keys.
   Customer a("Joe", "Morrison", "11 St. Calgary.", "(403)-1111-123333");
   Customer b("Jack", "Lewis", "12 St. Calgary.", "(403)-1111-123334");
   Customer c("Tim", "Hardy", "13 St. Calgary.", "(403)-1111-123335");
   lt.insert(Pair<int, Customer> (8002, a));
   lt.insert(Pair<int, Customer> (8004,c));
   lt.insert(Pair<int, Customer> (8001,b));
   assert(lt.size() == 3);
   lt.remove(8004);
   assert(lt.size() == 2);
   cout << "\nPrinting table after inserting 3 new keys and 1 removal...\n";
   print(lt);

   // Pretend that a user is trying to look up customers info.
    cout << "\nLet's look up some names ...\n";
   try_to_find(lt, 8001);
   try_to_find(lt, 8000);


   // test Iterator
   cout << "\nTesing and using  iterator ...\n";
   LookupTable<int,Customer>::Iterator it = lt.begin();
   cout <<"\nThe first node contains: " <<*it <<endl;


   while (!it) {
     cout <<++it << endl;
   }


   //test copying
   lt.go_to_first();
   lt.step_fwd();
   LookupTable<keyType,datumType> clt(lt);
   assert(strcmp(clt.cursor_datum().getFname(),"Joe")==0);


   cout << "\nTest copying: keys should be 8001, and 8002\n";
   print(clt);
   lt.remove(8002);
   //Assignment operator check.
   clt= lt;
    cout << "\nTest assignment operator: key should be 8001\n";
   print(clt);


   //Wipe out the entries in the table.
   lt.make_empty();
   cout << "\nPrinting table for the last time: Table should be empty...\n";
   print(lt);
```

```
   cout << "***----Finished tests on Customers Lookup Table <not template>-----***\n";
   cout << "PRESS RETURN TO CONTINUE.";
   cin.get();

 }



template <typename keyType, typename datumType>
void test_String()


 // creating lookuptable for Mystring objects
 {
   cout<<"\nCreating and testing LookupTable <int, Mystring> .....\n";
   LookupTable<keyType, datumType> lt;


   // Insert using new keys.


   Mystring a("I am an ENEL-409 student.");
   Mystring b("C++ is a powerful language for engineers but it's not easy.");
   Mystring c ("Winter 2004");


   lt.insert(Pair<int, Mystring> (8002,a));
   lt.insert(Pair<int, Mystring> (8001,b));
   lt.insert(Pair<int, Mystring> (8004,c));
   //assert(lt.size() == 3);
   //lt.remove(8004);
   //assert(lt.size() == 2);
   cout << "\nPrinting table after inserting 3 new keys and  and 1 removal...\n";
   print(lt);


   // Pretend that a user is trying to look up customers info.

   cout << "\nLet's look up some names ...\n";
   try_to_find(lt, 8001);
   try_to_find(lt, 8000);
   // test Iterator
   typename LookupTable<keyType ,datumType>::Iterator it = lt.begin();
   cout <<"\nThe first node contains: " <<*it <<endl;


   while (!it) {
     cout <<++it << endl;
   }


   //test copying
   lt.go_to_first();
   lt.step_fwd();
   LookupTable<keyType, datumType>  clt(lt);
   assert(strcmp(clt.cursor_datum().c_str(),"I am an ENEL-409 student.")==0);


   cout << "\nTest copying: keys should be 8001, and 8002\n";
   print(clt);
   lt.remove(8002);
   //Assignment operator check.
   clt= lt;
    cout << "\nTest assignment operator: key should be 8001\n";
   print(clt);
```

```
  // Wipe out the entries in the table.
  lt.make_empty();
  cout << "\nPrinting table for the last time: Table should be empty ...\n";
  print(lt);


  cout << "***----Finished Lab 4 tests on <int> <Mystring>-----***\n";
  cout << "PRESS RETURN TO CONTINUE.";
  cin.get();
}









template <typename keyType, typename datumType>
void test_integer()


 //creating look table of integers


 {
  cout<<"\nCreating and testing LookupTable <int, int> .....\n";
  LookupTable<keyType, datumType> lt;


  // Insert using new keys.
  lt.insert(Pair<int, int>(8002,9999));
  lt.insert(Pair<int, int>(8001,8888));
  lt.insert(Pair<int, int>(8004,8888));
  assert(lt.size() == 3);
  lt.remove(8004);
  assert(lt.size() == 2);
  cout << "\nPrinting table after inserting 3 new keys and  and 1 removal...\n";
  print(lt);


  // Pretend that a user is trying to look up customers info.

  cout << "\nLet's look up some names ...\n";
  try_to_find(lt, 8001);
  try_to_find(lt, 8000);


  // test Iterator
  typename LookupTable<keyType ,datumType>::Iterator it = lt.begin();


  while (!it) {
    cout <<++it << endl;

  }


  //test copying
  lt.go_to_first();
  lt.step_fwd();
  LookupTable<keyType, datumType> clt(lt);
  assert(clt.cursor_datum()== 9999);


  cout << "\nTest copying: keys should be 8001, and 8002\n";
  print(clt);
  lt.remove(8002);
  //Assignment operator check.
```

```
  clt= lt;
   cout << "\nTest assignment operator: key should be 8001\n";
  print(clt);



  // Wipe out the entries in the table.
  lt.make_empty();
  cout << "\nPrinting table for the last time: Table should be empty ...\n";
  print(lt);


  cout << "***----Finished Lab 4 tests on <int> <int>-----***\n";

}
```

Output

```
Printing table for the last time: Table should be empty...
  Table is EMPTY.
***----Finished tests on Customers Lookup Table <not template>-----***
PRESS RETURN TO CONTINUE.

Creating and testing LookupTable <int, Mystring> .....

Printing table after inserting 3 new keys and  and 1 removal...
8001  C++ is a powerful language for engineers but it's not easy.
8002  I am an ENEL-409 student.
8004  Winter 2004

Let's look up some names ...

Found key:8001  C++ is a powerful language for engineers but it's not easy.
Sorry, I couldn't find key: 8000 in the table.

The first node contains: C++ is a powerful language for engineers but it's not easy.
C++ is a powerful language for engineers but it's not easy.
I am an ENEL-409 student.
Winter 2004

Test copying: keys should be 8001, and 8002
8001  C++ is a powerful language for engineers but it's not easy.
8002  I am an ENEL-409 student.
8004  Winter 2004

Test assignment operator: key should be 8001
8001  C++ is a powerful language for engineers but it's not easy.
8004  Winter 2004

Printing table for the last time: Table should be empty ...
  Table is EMPTY.
***----Finished Lab 4 tests on <int> <Mystring>-----***
PRESS RETURN TO CONTINUE.

Creating and testing LookupTable <int, int> .....

Printing table after inserting 3 new keys and  and 1 removal...
8001  8888
8002  9999

Let's look up some names ...

Found key:8001  8888
Sorry, I couldn't find key: 8000 in the table.
8888
9999

Test copying: keys should be 8001, and 8002
8001  8888
8002  9999

Test assignment operator: key should be 8001
8001  8888

Printing table for the last time: Table should be empty ...
  Table is EMPTY.
***----Finished Lab 4 tests on <int> <int>-----***


Program terminated successfully.

(base) alyfarouz@mac exC %
```