Names: Neha Parmar, Shahed Issa

Course Name: Principles of Software Design

Lab Section: B01

Course Code: ENSF 480

Assignment Number: Lab-3

Submission Date and Time: 30/09/2024

Exercise A:

Circle.cpp

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

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

//copy constructor
Circle::Circle(const Circle& other) : Shape(other), radius(other.radius) {

}

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

//destructor:
Circle::~Circle() {
}

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

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

double Circle::get_radius() const {
    return radius;
}

void Circle::set_radius(double r) {
    radius = r;
}

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

Circle.h

```
#ifndef CIRCLE_H
#define CIRCLE_H

#include "Shape.h"
#include "Point.h"

class Circle : virtual public Shape {
protected:
    double radius;

public:
    Circle(double x, double y, double radius, const char* name);
    Circle(const Circle& other);
    Circle& operator=(const Circle& rhs);

    ~Circle();

    double area() const;

    double perimeter() const;

    double get_radius() const;
    void set_radius(double radius);

    virtual void display() const;
};

#endif
```

CurveCut.cpp

```
#include "CurveCut.h"
#include <iostream>
#include <cmath>

using namespace std;

// Constructor
CurveCut::CurveCut(double x, double y, double side_a, double side_b, double
radius, const char* name)
    : Shape(Point(x,y), name), Rectangle(Point(x,y), name, side_a, side_b),
Circle(x, y, radius, name) {
    if (radius > side_a || radius > side_b) {
        cerr << "Error: Radius must be less than or equal to the smaller of
width and length." << endl;
        exit(1);
    }
}

// Copy Constructor
CurveCut::CurveCut(const CurveCut& other) : Rectangle(other), Circle(other),
Shape(other.getOrigin(), other.getName()) {}
```

```
// Assignment Operator
CurveCut& CurveCut::operator=(const CurveCut& rhs) {
    if (this != &rhs) {
        Rectangle::operator=(rhs);
        Circle::operator=(rhs);
    }
    return *this;
}

CurveCut::~CurveCut() {}

double CurveCut::area() const {
    double rectArea = Rectangle::area();
    double circleCutArea = M_PI * get_radius() * get_radius() * 0.25;
    return (rectArea - circleCutArea);
}

double CurveCut::perimeter() const {
    double rectanglePerimeter = Rectangle::perimeter();
    double circleCutPerimeter = Circle::perimeter() * 0.25;
    return (rectanglePerimeter - (2 * get_radius()) + circleCutPerimeter);
}

void CurveCut::display() const {
    cout << "CurveCut Name: " << getName() << endl;
    cout << "X-coordinate: " << getOrigin().getx() << endl;
    cout << "Y-coordinate: " << getOrigin().gety() << endl;
    cout << "Width: " << get_side_b() << endl;
    cout << "Length: " << get_side_a() << endl;
    cout << "Radius of the cut: " << get_radius() << endl;
}
```

CurveCut.h

```
#ifndef CURVECUT_H
#define CURVECUT_H

#include "Rectangle.h"
#include "Circle.h"

class CurveCut : public Rectangle, public Circle {
public:
    CurveCut(double x, double y, double side_a, double side_b, double
radius, const char* name);

    CurveCut(const CurveCut& other);

    CurveCut& operator=(const CurveCut& rhs);

    ~CurveCut();

    double area() const;
```

```
    double perimeter() const;

    void display() const;
};

#endif // CURVECUT_H
```

GraphicsWorld.cpp

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

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

void GraphicsWorld::run() {
    std::cout << "made by Neha Parmar and Shahed Issa";

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

    // Change 0 to 1 to test Square
    #if 1
    std::cout << "\n\nTesting Functions in class Square:" << std::endl;
    Point p1(5, 7);  // Create a Point object for Square constructor
    Square s(p1, "SQUARE - S", 12);
    s.display();
    #endif // end of block to test Square

    // Change 0 to 1 to test Rectangle
    #if 1
    std::cout << "\nTesting Functions in class Rectangle:\n";
    Point p2(5, 7);  // Create Point object for Rectangle constructor
    Rectangle a(p2, "RECTANGLE A", 12, 15);
    a.display();

    Point p3(16, 7);  // Create Point object for another Rectangle
    Rectangle b(p3, "RECTANGLE B", 8, 9);
```

```
    b.display();

    double d = a.distance(b);
    std::cout << "\nDistance between rectangle a and b is: " << d <<
std::endl;

    Rectangle rec1 = a;
    rec1.display();

    std::cout << "\nTesting assignment operator in class Rectangle:" <<
std::endl;
    Point p4(3, 4);  // Create Point object for another Rectangle
    Rectangle rec2(p4, "RECTANGLE rec2", 11, 7);
    rec2.display();
    rec2 = a;

    a.set_side_b(200);
    a.set_side_a(100);

    std::cout << "\nExpected to display the following values for object
rec2: " << std::endl;
    std::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";
    std::cout << "\nIf it doesn't, there is a problem with your assignment
operator.\n" << std::endl;
    rec2.display();

    std::cout << "\n\nTesting copy constructor in class Rectangle:" <<
std::endl;
    Rectangle rec3(a);
    rec3.display();

    a.set_side_b(300);
    a.set_side_a(400);

    std::cout << "\n\nExpected to display the following values for object
rec3: " << std::endl;
    std::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";
    std::cout << "\nIf it doesn't, there is a problem with your copy
constructor.\n" << std::endl;
    rec3.display();
    #endif

    // Change 0 to 1 to test using array of pointers and polymorphism
    #if 0
```

```
    std::cout << "\nTesting array of pointers and polymorphism:" <<
std::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 pointers and polymorphism

    #if 1
    cout << "\n\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.Rectangle::getName() << " is: " <<
rc.area() << endl;
    cout << "The perimeter of " << rc.Rectangle::getName() << " is: " <<
rc.perimeter() << endl;

    d = rc.Rectangle::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 << "The area of " << sh[0]->getName() << " is: " << sh[0]->area()
<< endl;
    cout << "The perimeter of " << sh[0]->getName() << " is: " <<
sh[0]->perimeter() << endl;

    sh[1]->display();
    cout << "The area of " << sh[1]->getName() << " is: " << sh[1]->area()
<< endl;
    cout << "The perimeter of " << sh[1]->getName() << " is: " <<
sh[1]->perimeter() << endl;

    sh[2]->display();
```

```
    cout << "The area of " << sh[2]->getName() << " is: " << sh[2]->area()
<< endl;
    cout << "The perimeter of " << sh[2]->getName() << " is: " <<
sh[2]->perimeter() << endl;

    sh[3]->display();
    cout << "The area of " << sh[3]->getName() << " is: " << sh[3]->area()
<< endl;
    cout << "The perimeter of " << sh[3]->getName() << " is: " <<
sh[3]->perimeter() << endl;

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

    // Testing assignment operator in class CurveCut:
    cout << "\n\nTesting assignment operator in class CurveCut:" << endl;
    CurveCut cc2(2, 5, 10, 12, 9, "CurveCut cc2");
    cc2.display();
    cc2 = rc;
    cc2.display();
#endif


}
```

GraphicsWorld.h

```
#ifndef GRAPHICS_WORLD_H
#define GRAPHICS_WORLD_H

#include "Point.h"
#include "Square.h"
#include "Rectangle.h"
#include "Shape.h"
#include "Circle.h"
#include "CurveCut.h"


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

#endif
```

Point.cpp

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

```
int Point::count = 1001;

Point::Point(double xCoord, double yCoord) : x(xCoord), y(yCoord) {
    id = count++;
}

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

double Point::gety() const {
    return y;
}

void Point::setx(double xCoord) {
    x = xCoord;
}

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

void Point::display() const {
    cout << "X-coordinate: " << x << endl;
    cout << "Y-coordinate: " << y << endl;
}

int Point::counter() {
    return count - 1001;
}

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

double Point::distance(const Point& p1, const Point& p2) {
    return sqrt((p1.x - p2.x) * (p1.x - p2.x) + (p1.y - p2.y) * (p1.y -
p2.y));
}
```

Point.h

```
#ifndef POINT_H
#define POINT_H

class Point {
private:
    static int count;
    int id;
    double x, y;
```

```
public:
    Point(double xCoord, double yCoord);

    void display() const;

    static int counter();

    double distance(const Point& other) const;

    static double distance(const Point& p1, const Point& p2);

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

    // Setters
    void setx(double xCoord);
    void sety(double yCoord);
};

#endif
```

Rectangle.cpp

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

Rectangle::Rectangle(const Point& p, const char* name, double side_a, double
side_b)
    : Square(p, name, side_a), side_b(side_b), Shape(p, name) {}

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

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

Rectangle::~Rectangle() {}

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

double Rectangle::perimeter() const {
```

```
    return 2 * (get_side_a() + side_b);
}

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

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

void Rectangle::display() const {
    cout << "Rectangle Name: " << getName() << endl;
    cout << "X-coordinate: " << getOrigin().getx() << endl;
    cout << "Y-coordinate: " << getOrigin().gety() << endl;
    cout << "Width (Side a): " << get_side_a() << endl;
    cout << "Height (Side b): " << side_b << endl;
    cout << "Area: " << area() << endl;
    cout << "Perimeter: " << perimeter() << endl;
}
```

Rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include "Square.h"

class Rectangle : public Square {
private:
    double side_b;

public:
    Rectangle(const Point& p, const char* name, double side_a, double
side_b);
    Rectangle(const Rectangle& other);
    Rectangle& operator=(const Rectangle& rhs);

    ~Rectangle();

    double area() const;

    double perimeter() const;

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

    void display() const;
};

#endif // RECTANGLE_H
```

Shape.cpp

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

//contructor
Shape::Shape(const Point& p, const char* name) : origin(p) {
    shapeName = new char[strlen(name) + 1];
    strcpy(shapeName, name);
}

//Copy contructor
Shape::Shape(const Shape& other) : origin(other.origin){
    shapeName = new char[strlen(other.shapeName)+1];
    strcpy(shapeName, other.shapeName);
}

//assignment operator
Shape& Shape::operator=(const Shape& rhs){
    if (this != &rhs){
        origin = rhs.origin;
        delete[] shapeName;

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

    }
}

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


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

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

void Shape::display() const {
    cout << "Shape Name: " << shapeName << endl;
    cout << "X-coordinate: " << origin.getx() << endl;
    cout << "Y-coordinate: " << origin.gety() << endl;
}


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

```
double Shape::distance(const Shape& shape1, const Shape& shape2) {
    return Point::distance(shape1.origin, shape2.origin);

}

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

Shape.h

```
#ifndef SHAPE_H
#define SHAPE_H

#include "Point.h"

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

public:
    Shape(const Point& p, const char* name);
    Shape(const Shape& other);
    Shape& operator=(const Shape& rhs);

    ~Shape();

    const Point& getOrigin() const;

    const char* getName() const;

    virtual void display() const;

    virtual double distance(const Shape& other) const;
    static double distance(const Shape& shape1, const Shape& shape2);

    void move(double dx, double dy);
    virtual double area() const =0 ;
    virtual double perimeter() const=0;
};

#endif
```

Square.cpp

```
#include "Square.h"
#include <iostream>
```

```
#include <cstring>
#include <cmath>
using namespace std;

//constructor
Square::Square(const Point& p, const char* name, double side_a)
    : Shape(p, name), side_a(side_a) {}

//copy constrcutor
Square::Square(const Square& other) : Shape(other), side_a(other.side_a) {

}

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

//destructor:
Square::~Square() {
}

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

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

double Square::get_side_a() const {
    return side_a;
}

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

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

Square.h

```
#ifndef SQUARE_H
#define SQUARE_H

#include "Shape.h"
#include "Point.h"

class Square : virtual public Shape {
protected:
    double side_a;

public:
    Square(const Point& p, const char* name, double side_a);
    Square(const Square& other);
    Square& operator=(const Square& rhs);

    ~Square();

    double area() const;

    double perimeter() const;

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

    virtual void display() const;
};

#endif
```

Output:

```
TERMINAL
            ^~
PS C:\Users\nehap\Desktop\ENSF-480\LAB3\ExerciseA> g++ -mconsole -o GraphicsWorld GraphicsW
rcle.cpp CurveCut.cpp
PS C:\Users\nehap\Desktop\ENSF-480\LAB3\ExerciseA> ./GraphicsWorld
made by Neha Parmar and Shahed Issa
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
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
Width (Side a): 12
Height (Side b): 15
Area: 180
Perimeter: 54
Rectangle Name: RECTANGLE B
X-coordinate: 16
Y-coordinate: 7
Width (Side a): 8
Height (Side b): 9
Area: 72
Perimeter: 34

Distance between rectangle a and b is: 11
Rectangle Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Width (Side a): 12
Height (Side b): 15
Area: 180
Perimeter: 54
```

```
Testing assignment operator in class Rectangle:
Rectangle Name: RECTANGLE rec2
X-coordinate: 3
Y-coordinate: 4
Width (Side a): 11
Height (Side b): 7
Area: 77
Perimeter: 36

Expected to display the following values for object 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
Width (Side a): 12
Height (Side b): 15
Area: 180
Perimeter: 54


Testing copy constructor in class Rectangle:
Rectangle Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Width (Side a): 100
Height (Side b): 200
Area: 20000
Perimeter: 600
```

```
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
Y-coordinate: 7
Width (Side a): 100
Height (Side b): 200
Area: 20000
Perimeter: 600


Testing Functions in class Circle:
Shape Name: CIRCLE C
X-coordinate: 3
Y-coordinate: 5
Radius: 9
Area: 254.469
Perimeter: 56.5487
The area of CIRCLE C is: 254.469
The perimeter of CIRCLE C is: 56.5487

The distance between rectangle a and circle c is: 2.82843
CurveCut Name: CurveCut rc
X-coordinate: 6
Y-coordinate: 5
Width: 12
Length: 10
Radius of the cut: 9
The area of CurveCut rc is: 56.3827
The perimeter of CurveCut rc is: 40.1372
```

```
The distance between rc and c is: 3
Square Name: SQUARE - S
X-coordinate: 5
Y-coordinate: 7
Side a: 12
Area: 144
Perimeter: 48
The area of SQUARE - S is: 144
The perimeter of SQUARE - S is: 48
Rectangle Name: RECTANGLE A
X-coordinate: 5
Y-coordinate: 7
Width (Side a): 400
Height (Side b): 300
Area: 120000
Perimeter: 1400
The area of RECTANGLE A is: 120000
The perimeter of RECTANGLE A is: 1400
Shape Name: CIRCLE C
X-coordinate: 3
Y-coordinate: 5
Radius: 9
Area: 254.469
Perimeter: 56.5487
The area of CIRCLE C is: 254.469
The perimeter of CIRCLE C is: 56.5487
CurveCut Name: CurveCut rc
X-coordinate: 6
Y-coordinate: 5
Width: 12
Length: 10
Radius of the cut: 9
The area of CurveCut rc is: 56.3827
The perimeter of CurveCut rc is: 40.1372


Testing copy constructor in class CurveCut:
CurveCut Name: CurveCut rc
X-coordinate: 6
Y-coordinate: 5
Width: 12
Length: 10
Radius of the cut: 9
```

```
Testing assignment operator in class CurveCut:
CurveCut Name: CurveCut cc2
X-coordinate: 2
Y-coordinate: 5
Width: 12
Length: 10
Radius of the cut: 9
CurveCut Name: CurveCut rc
X-coordinate: 6
Y-coordinate: 5
Width: 12
Length: 10
Radius of the cut: 9
```

Exercise B:

Iterator.cpp

```
// iterator.cpp
// ENSF 480 - Fall 2022 - Lab 3, Ex B
// Authors - Neha Parmar & Shahed Issa

#include <iostream>
#include <assert.h>
#include "mystring2.h"
#include "cstring"

using namespace std;

template <class T>
class Vector {
public:

  class VectIter{
    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<T> & 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++)
      if (strcmp(array[i], array[j]) > 0)
        swap(array[i], array[j]);
  }
}

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 *()
{
  return v -> array[index];
}
```

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

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

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

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()
{

 Vector<int> x(3);
 x[0] = 999;
 x[1] = -77;
 x[2] = 88;

 Vector<int>::VectIter iter(x);

 cout << "\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

#if 1
      cout << "\nTesting an <int> Vector: " << endl;

      cout << "Testing sort";
      x.ascending_sort();

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

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

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

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

      cout << endl;

      cout << "\nTesting a <Mystring> Vector: " << endl;
      Vector<Mystring> y(3);
      y[0] = "Bar";
      y[1] = "Foo";
      y[2] = "All";;

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

      cout << "Testing sort";
      y.ascending_sort();

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

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

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

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

      cout << endl; cout << "\nTesting 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 << "Testing sort";
      z.ascending_sort();

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

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

      return 0;
}
```

mystring2.cpp

```
//  mystring2.cpp
// ENSF 480 - Lab 3
// M. Moussavi
#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);
}

int Mystring::isNotEqual (const Mystring& s)const
{
```

```
  return (strcmp(charsM, s.charsM)!= 0);
}

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


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

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

ostream &operator<<(ostream &os, const Mystring &source)
{
  os << source.charsM;
  return os;
}

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

bool Mystring::operator<(const Mystring& other) const {
    return strcmp(charsM, other.charsM) < 0;
}
```

mystring2.h

```
//File: mystring2.h
// ENSF 480 - Lab 3


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

  int isLessThan (const Mystring& s)const;
```

```
  // REQUIRES: s refers to an object of class Mystring
  // PROMISES: retruns true if charsM is less than s.charsM.

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

  int isNotEqual(const Mystring& s)const;
  // REQUIRES: s refers to an object of class Mystring
  // PROMISES: retruns true if charsM is not equal s.charsM.
  bool operator>(const Mystring& other) const;
  bool operator<(const Mystring& other) const;

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

  // PROMISES: if s points to NULL terminates the program.
};
#endif
```

Output:

```
PS C:\Users\nehap\Desktop\ENSF-480\LAB3> cd ExerciseB
PS C:\Users\nehap\Desktop\ENSF-480\LAB3\ExerciseB> ./Iterator

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
Prgram Terminated Successfully.
```

Exercise C:
lookupTable.h:

```
#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"

//    In this version of the LookupTable a new struct type called Pair
//    is introduced which represents a key/data pair.

template <class K, class D>
class LookupTable;

template <class K, class D>
struct Pair
{
  Pair(K keyA, D datumA) : key(keyA), datum(datumA) {}

  K key;
  D datum;
};

// LT_Node class
template <class K, class D>
class LT_Node
{
  friend class LookupTable<K, D>;

private:
  Pair<K, D> pairM;
  LT_Node *nextM;


  LT_Node(const Pair<K, D> &pairA, LT_Node *nextA) : pairM(pairA),
nextM(nextA) {}
};

//LookupTable class
template <class K, class D>
class LookupTable
{
```

```
public:
  // Nested class
  class Iterator
  {
    friend class LookupTable;
    LookupTable *LT;

  public:
    Iterator() : LT(0) {}
    Iterator(LookupTable &x) : LT(&x) {}
    const D &operator*();
    const D &operator++();
    const D &operator++(int);
    int operator!();

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

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

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

  LookupTable &begin();

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

  int cursor_ok() const {
    return cursorM != 0;
  }
  // 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 K &cursor_key() const;
  // REQUIRES: cursor_ok()
  // PROMISES: Returns key of key/datum pair to which cursor is attached.

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

  void insert(const Pair<K, D> &pairA);
  // 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 K &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 K &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.

  template <class K1, class D1> // i dont know why this is needed but it
doesn't compile without it
  friend ostream &operator<<(ostream &os, const LookupTable &lt);

private:
  int sizeM;
  LT_Node<K, D> *headM;
  LT_Node<K, D> *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

// Implementations
template <class K, class D>
LookupTable<K, D>::LookupTable(const LookupTable &source) { copy(source); }

template <class K, class D>
LookupTable<K, D>::~LookupTable() { destroy(); }
```

```
template <class K, class D>
LookupTable<K, D> &LookupTable<K, D>::operator=(const LookupTable &rhs)
{
  if (this != &rhs)
  {
    destroy();
    copy(rhs);
  }
  return *this;
}

template <class K, class D>
LookupTable<K, D> &LookupTable<K, D>::begin()
{
  cursorM = headM;
  return *this;
}

template <class K, class D>
const K &LookupTable<K, D>::cursor_key() const
{
  assert(cursor_ok());
  return cursorM->pairM.key;
}

template <class K, class D>
const D &LookupTable<K, D>::cursor_datum() const
{
  assert(cursor_ok());
  return cursorM->pairM.datum;
}

template <class K, class D>
void LookupTable<K, D>::insert(const Pair<K, D> &pairA)
{
  // Add new node at head if list is empty or key is less than head key
  if (headM == 0 || pairA.key < headM->pairM.key) {
    headM = new LT_Node(pairA, headM);
    sizeM++;
  }
  // If key is equal to head key, update datum
  else if (pairA.key == headM->pairM.key)
    headM->pairM.datum = pairA.datum;


  else {
    LT_Node<K, D> *before = headM;
    LT_Node<K, D> *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(pairA, before->nextM);
      sizeM++;
    }
  }
}

template <class K, class D>
void LookupTable<K, D>::remove(const K &keyA)
{

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

  LT_Node<K, D> *doomed_node = 0;
  if (keyA == headM->pairM.key) {
    doomed_node = headM;
    headM = headM->nextM;
    sizeM--;
  }
  else {
    LT_Node<K, D> *before = headM;
    LT_Node<K, D> *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;
}

template <class K, class D>
void LookupTable<K, D>::find(const K &keyA) {
  LT_Node<K, D> *ptr = headM;
  while (ptr != NULL && ptr->pairM.key != keyA)
  {
    ptr = ptr->nextM;
  }

  cursorM = ptr;
}

template <class K, class D>
void LookupTable<K, D>::go_to_first() {
  cursorM = headM;
}
```

```
template <class K, class D>
void LookupTable<K, D>::step_fwd() {
  assert(cursor_ok());
  cursorM = cursorM->nextM;
}

template <class K, class D>
void LookupTable<K, D>::make_empty() {
  destroy();
  sizeM = 0;
  cursorM = 0;
}

template <class K, class D>
void LookupTable<K, D>::destroy() {
  LT_Node<K, D> *ptr = headM;
  while (ptr != NULL)
  {
    headM = headM->nextM;
    delete ptr;
    ptr = headM;
  }
  cursorM = NULL;
  sizeM = 0;
}

template <class K, class D>
void LookupTable<K, D>::copy(const LookupTable<K, D> &source) {

  headM = 0;
  cursorM = 0;

  if (source.headM == 0)
    return;

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

template <class K, class D>
ostream &operator<<(ostream &os, const LookupTable<K, D> &lt) {
  if (lt.cursor_ok())
    os << lt.cursor_key() << "  " << lt.cursor_datum();
  else
    os << "Not Found.";

  return os;
}

template <class K, class D>
const D &LookupTable<K, D>::Iterator::operator*() {
  assert(LT->cursor_ok());
```

```
  return LT->cursor_datum();
}

template <class K, class D>
const D &LookupTable<K, D>::Iterator::operator++() {
  assert(LT->cursor_ok());
  const D &x = LT->cursor_datum();
  LT->step_fwd();
  return x;
}

template <class K, class D>
const D &LookupTable<K, D>::Iterator::operator++(int) {
  assert(LT->cursor_ok());

  LT->step_fwd();
  return LT->cursor_datum();
}

template <class K, class D>
int LookupTable<K, D>::Iterator::operator!() {
  return (LT->cursor_ok());
}
```

mainLab3ExC.cpp:

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

template <class K, class D>
void print(LookupTable<K, D> &lt);

template <class K, class D>
void try_to_find(LookupTable<K, D> &lt, K key);

void test_Customer();
// Uncomment the following function calls when ready to test template class
LookupTable
void test_String();
void test_integer();

int main() {

  // create and test a lookup table with an integer key value and Customer
datum
  test_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();
```

```
  // 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();

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

  return 0;
}

template <class K, class D>
void print(LookupTable<K, D> &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 <class K, class D>
void try_to_find(LookupTable<K, D> &lt, K 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";
}

void test_Customer() {
  cout << "\nCreating and testing Customers Lookup Table <not
template>-...\n";
  LookupTable<int, Customer> 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(8002, a));
  lt.insert(Pair(8004, c));
  lt.insert(Pair(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 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();
}

// Uncomment and modify the following funciton when ready to test
LookupTable<int,Mystring>

void test_String() {
  cout << "\nCreating and testing LookupTable <int, Mystring> .....\n";
  LookupTable<int, Mystring> 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
  LookupTable<int, Mystring>::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 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();
}

// Uncomment and modify the following funciton when ready to test
LookupTable<int,int>

void test_integer() {
  cout << "\nCreating and testing LookupTable <int, int> .....\n";
  LookupTable<int, int> 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
  LookupTable<int, int>::Iterator it = lt.begin();

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

  // test copying
  lt.go_to_first();
  lt.step_fwd();
  LookupTable 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";
}
```

```
shahe@Shaheds_Laptop ~/ensf480/lab3
$ g++ -Wall -o exc customer.cpp mainLab3ExC.cpp mystring2.cpp

shahe@Shaheds_Laptop ~/ensf480/lab3
$ ./exc

Creating and testing Customers Lookup Table <not template>-...

Printing table after inserting 3 new keys and 1 removal...
8001  Nmae: Jack Lewis. Address: 12 St. Calgary.. Phone:: (403)-1111-123334
8002  Nmae: Joe Morrison. Address: 11 St. Calgary.. Phone:: (403)-1111-123333

Let's look up some names ...

Found key:8001  Nmae: Jack Lewis. Address: 12 St. Calgary.. Phone:: (403)-1111-1
23334
Sorry, I couldn't find key: 8000 in the table.

Tesing and using  iterator ...

The first node contains: Nmae: Jack Lewis. Address: 12 St. Calgary.. Phone:: (40
3)-1111-123334
Nmae: Jack Lewis. Address: 12 St. Calgary.. Phone:: (403)-1111-123334
Nmae: Joe Morrison. Address: 11 St. Calgary.. Phone:: (403)-1111-123333

Test copying: keys should be 8001, and 8002
8001  Nmae: Jack Lewis. Address: 12 St. Calgary.. Phone:: (403)-1111-123334
8002  Nmae: Joe Morrison. Address: 11 St. Calgary.. Phone:: (403)-1111-123333

Test assignment operator: key should be 8001
8001  Nmae: Jack Lewis. Address: 12 St. Calgary.. Phone:: (403)-1111-123334

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 e
asy.
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
```

```
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.


shahe@Shaheds_Laptop ~/ensf480/lab3
```