**Name (s): Kaumil Patel**

**Course Name:** Principles of Software Design

**Lab Section:** B01

**Course Code:** ENSF 480

**Assignment Number:** Lab 3

**Submission Date and Time:** Oct 14 2021

**Exercise A - Multiple-Inheritance (10 marks)**

Exercise B - Inheritance

Written by: Kaumil Patel

Testing will commence now

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

The distance between m and n is: 3

Expected second version of the distance function also print: 3

The distance between m and n is again: 3

Testing Functions in class Square:

Square Name: SQUARE - S

X-coordinate: 5.00

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 assignment operator.

Rectangle Name: RECTANGLE A

X-coordinate: 5.00

Y-coordinate: 7.00

Side a: 100.00

Side b: 200.00

Area: 20000.00

Perimeter: 600.00

Testing array of pointers and polymorphism:

Square Name: SQUARE - S

X-coordinate: 5.00

Y-coordinate: 7.00

Side a: 12.00

Area: 144.00

Perimeter: 48.00

Rectangle Name: RECTANGLE B

X-coordinate: 16.00

Y-coordinate: 7.00

Side a: 8.00

Side b: 9.00

Area: 72.00

Perimeter: 34.00

Rectangle Name: RECTANGLE A

X-coordinate: 5.00

Y-coordinate: 7.00

Side a: 12.00

Side b: 15.00

Area: 180.00

Perimeter: 54.00

Rectangle Name: RECTANGLE A

X-coordinate: 5.00

Y-coordinate: 7.00

Side a: 100.00

Side b: 200.00

Area: 20000.00

Perimeter: 600.00

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: 2.83CurveCut 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: 56.38the perimeter of CurveCut rc is: 29.86

The distance between rc and c is: 3.00Square 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 - Sis: 144.00

the perimeter of SQUARE - S is: 48.00Rectangle 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 Ais: 120000.00

the perimeter of SQUARE - S is: 1400.00Circle Name: CIRCLE C

X-coordinate: 3.00

Y-coordinate: 5.00

Radius: 9.00

Area: 254.47

Perimeter: 56.55

the area of CIRCLE Cis: 254.47

the circumference of CIRCLE C is: 56.55CurveCut 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 rcis: 56.38

the perimeter of CurveCut rc is: 29.86

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

Process finished with exit code 0

/\*  
\* File Name: square.h  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/  
  
#ifndef EXERCISE\_A\_CIRCLE\_H  
#define EXERCISE\_A\_CIRCLE\_H  
  
  
#include "shape.h"  
  
class Circle :virtual public Shape {  
public:  
 Circle(const double &x, const double &y, const double &sideLength, const char \*name);  
  
 const double area()const override;  
  
 const double perimeter()const override;  
  
 void display() override;  
  
 void setradius(const double &radius);  
  
 const double &getradius() const;  
  
  
protected:  
 double radius;  
};  
  
  
#endif //EXERCISE\_A\_CIRCLE\_H

/\*  
\* File Name: square.cpp  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/  
  
#include "iostream"  
#include "iomanip"  
#include <cmath>  
  
using namespace std;  
  
#include "circle.h"  
  
Circle::Circle(const double &x, const double &y, const double &radius, const char \*name) : Shape(x, y, name),  
 radius(radius) {}  
  
void Circle::display() {  
 cout << setprecision(2) << fixed;  
 cout << "Circle Name: " << shapeName << endl;  
 cout << "X-coordinate: " << origin.getx() << endl;  
 cout << "Y-coordinate: " << origin.gety() << endl;  
 cout << "Radius: " << radius << endl;  
 cout << "Area: " << area() << endl;  
 cout << "Perimeter: " << perimeter() << endl;  
}  
  
const double Circle::perimeter() const{  
 return M\_PI \* 2 \* radius;  
}  
  
const double Circle::area()const {  
 return M\_PI \* radius \* radius;  
}  
  
void Circle::setradius(const double &radius) {  
 this->radius = radius;  
}  
  
const double &Circle::getradius() const {  
 return radius;  
}

/\*  
\* File Name: curveCut.h  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/  
  
#ifndef EXERCISE\_A\_CURVECUT\_H  
#define EXERCISE\_A\_CURVECUT\_H  
  
#include "circle.h"  
#include "rectangle.h"  
  
class CurveCut : public Circle, public Rectangle {  
public:  
 CurveCut(const double &x, const double &y, const double &sideA, const double &sideB, const double &radius, const char\* name);  
 CurveCut(const CurveCut &rhs);  
 CurveCut& operator=(const CurveCut &rhs);  
  
 const double area()const override;  
  
 const double perimeter()const override;  
  
 void display() override;  
  
  
};  
  
  
#endif //EXERCISE\_A\_CURVECUT\_H

/\*  
\* File Name: curveCut.cpp  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/  
  
#include "iostream"  
#include "iomanip"  
#include <cassert>  
  
using namespace std;  
  
#include "curveCut.h"  
  
CurveCut::CurveCut(const double &x, const double &y, const double &sideA, const double &sideB, const double &radius,  
 const char \*name) : Shape(x, y, name), Square(x, y, sideA, name),  
 Rectangle(x, y, sideA, sideB, name),  
 Circle(x, y, radius, name) {  
 if (radius > sideA || radius > sideB) {  
 cout << "Error: radius is longer than side lengths" << endl;  
 }  
 assert(radius <= sideA && radius <= sideB);  
}  
  
CurveCut::CurveCut(const CurveCut &rhs) : Shape(rhs.origin.getx(), rhs.origin.gety(), rhs.shapeName),  
 Square(rhs.origin.getx(), rhs.origin.gety(), rhs.side\_a, rhs.shapeName),  
 Rectangle(rhs.origin.getx(), rhs.origin.gety(), rhs.side\_a, rhs.side\_b,  
 rhs.shapeName),  
 Circle(rhs.origin.getx(), rhs.origin.gety(), rhs.radius, rhs.shapeName) {  
 if (radius > rhs.side\_a || radius > rhs.side\_b) {  
 cout << "Error: radius is longer than side lengths" << endl;  
 }  
 assert(radius <= rhs.side\_a && radius <= rhs.side\_b);  
}  
  
CurveCut &CurveCut::operator=(const CurveCut &rhs) {  
 if (this != &rhs) {  
 setName(rhs.shapeName);  
 origin.setx(rhs.origin.getx());  
 origin.sety(rhs.origin.gety());  
 side\_a = rhs.side\_a;  
 side\_b = rhs.side\_b;  
 radius = rhs.radius;  
 }  
 return \*this;  
}  
  
void CurveCut::display() {  
 cout << setprecision(2) << fixed;  
 cout << "CurveCut Name: " << shapeName << endl;  
 cout << "X-coordinate: " << origin.getx() << endl;  
 cout << "Y-coordinate: " << origin.gety() << endl;  
 cout << "Width: " << side\_a << endl;  
 cout << "Length: " << side\_b << endl;  
 cout << "Radius of the cut: " << radius << endl;  
}  
  
const double CurveCut::perimeter() const {  
 return Rectangle::perimeter() - Circle::perimeter() / 4.f;  
}  
  
const double CurveCut::area() const {  
 return Rectangle::area() - Circle::area() / 4.f;  
}

/\*  
\* File Name: rectangle.h  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021\*/  
  
#ifndef EXERCISE\_B\_RECTANGLE\_H  
#define EXERCISE\_B\_RECTANGLE\_H  
  
#include "square.h"  
  
class Rectangle :virtual public Square{  
public:  
 Rectangle(const double &x, const double &y, const double &sideA, const double &sideB, const char\* name);  
 void set\_side\_a(const double& i);  
 void set\_side\_b(const double& i);  
 void display() override;  
  
 const double perimeter()const override;  
  
 const double area()const override;  
  
// double distance(Rectangle &other);  
//  
// double distance(Rectangle &the\_shape, Rectangle &other);  
protected:  
 double side\_b;  
};  
  
  
#endif //EXERCISE\_B\_RECTANGLE\_H

/\*  
\* File Name: rectangle.cpp  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021\*/  
  
#include "iostream"  
#include "iomanip"  
  
using namespace std;  
  
#include "rectangle.h"  
  
Rectangle::Rectangle(const double &x, const double &y, const double &sideA, const double &sideB, const char\* name) : Shape(x, y, name), Square(x,y,sideA,name), side\_b(sideB) {}  
  
void Rectangle::set\_side\_a(const double &a) {  
 side\_a = a;  
  
}  
  
void Rectangle::set\_side\_b(const double &b) {  
 side\_b = b;  
}  
  
void Rectangle::display() {  
 cout << setprecision(2) << fixed;  
 cout << "Rectangle Name: " << shapeName << endl;  
 cout << "X-coordinate: " << origin.getx() << endl;  
 cout << "Y-coordinate: " << origin.gety() << endl;  
 cout << "Side a: " << side\_a << endl;  
 cout << "Side b: " << side\_b << endl;  
 cout << "Area: " << area() << endl;  
 cout << "Perimeter: " << perimeter() << endl;  
}  
  
const double Rectangle::perimeter() const{  
 return 2 \* side\_a + 2 \* side\_b;  
}  
  
const double Rectangle::area() const{  
 return side\_a \* side\_b;  
}  
  
//double Rectangle::distance(Rectangle &other) {  
// Point p1(origin.getx() + side\_a / 2, origin.gety() + side\_b / 2);  
// Point p2(other.origin.getx() + other.side\_a / 2, other.origin.gety() + other.side\_b / 2);  
// return p1.distance(p2);  
//}  
//  
//double Rectangle::distance(Rectangle &the\_shape, Rectangle &other) {  
// Point p1(the\_shape.origin.getx() + the\_shape.side\_a / 2, the\_shape.origin.gety() + the\_shape.side\_b / 2);  
// Point p2(other.origin.getx() + other.side\_a / 2, other.origin.gety() + other.side\_b / 2);  
// return p1.distance(p2);  
//}

/\*  
\* File Name: shape.h  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/  
  
#ifndef EXERCISE\_B\_SHAPE\_H  
#define EXERCISE\_B\_SHAPE\_H  
  
#include "point.h"  
  
class Shape {  
public:  
 Shape(const double &x, const double &y, const char \*name);  
  
 ~Shape();  
  
 Shape &operator=(const Shape &other);  
  
 const Point &getOrigin();  
  
 char \*getName();  
  
 void setName(const char \*name);  
  
 virtual void display() ;  
  
 const double distance(Shape &other);  
  
 static const double distance(Shape &the\_shape, Shape &other);  
  
 void move(const double& dx, const double& dy);  
  
 virtual const double area()const = 0;  
 virtual const double perimeter()const = 0;  
  
protected:  
 Point origin;  
 char \*shapeName;  
};  
  
  
#endif //EXERCISE\_B\_SHAPE\_H

/\*  
\* File Name: shape.cpp  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/  
  
#include "cstring"  
#include "iostream"  
#include "iomanip"  
  
using namespace std;  
  
#include "shape.h"  
  
Shape::Shape(const double &x, const double &y, const char \*name) : origin(x, y), shapeName(new char[strlen(name) + 1]) {  
 strcpy(this->shapeName, name);  
}  
  
Shape::~Shape() {  
 delete[] shapeName;  
}  
  
Shape &Shape::operator=(const Shape &other) {  
 if (this != &other) {  
 origin = other.origin;  
 delete[] shapeName;  
 shapeName = new char[strlen(other.shapeName) + 1];  
 strcpy(shapeName, other.shapeName);  
 }  
 return \*this;  
}  
  
void Shape::display() {  
 cout << setprecision(2) << fixed;  
 cout << "Shape Name: " << shapeName << endl;  
 cout << "X-coordinate: " << origin.getx() << endl;  
 cout << "Y-coordinate: " << origin.gety() << endl;  
}  
  
const double Shape::distance(Shape &other) {  
 return this->origin.distance(other.origin);  
}  
  
const double Shape::distance(Shape &the\_shape, Shape &other) {  
 return the\_shape.origin.distance(other.origin);  
}  
  
const Point &Shape::getOrigin() {  
 return origin;  
}  
  
char \*Shape::getName() {  
 return shapeName;  
}  
  
void Shape::setName(const char \*name) {  
 delete[] shapeName;  
 shapeName = new char[strlen(name) + 1];  
 strcpy(shapeName, name);  
}  
  
void Shape::move(const double& dx, const double& dy) {  
 origin.x += dx;  
 origin.y += dy;  
}

/\*  
\* File Name: square.h  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/  
  
#ifndef EXERCISE\_B\_SQUARE\_H  
#define EXERCISE\_B\_SQUARE\_H  
  
#include "point.h"  
#include "shape.h"  
  
class Square : virtual public Shape {  
public:  
 Square(const double &x, const double &y, const double &sideLength, const char \*name);  
  
 const double area()const override;  
  
 const double perimeter()const override;  
  
 void display() override ;  
  
// double distance(Square &other);  
//  
// static double distance(Square &the\_shape, Square &other);  
  
protected:  
 double side\_a;  
};  
  
  
#endif //EXERCISE\_B\_SQUARE\_H

/\*  
\* File Name: square.cpp  
\* Assignment: Lab 3 Exercise A  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/  
  
#include "iostream"  
#include "iomanip"  
  
using namespace std;  
  
#include "square.h"  
  
Square::Square(const double &x, const double &y, const double &sideLength, const char \*name) : Shape(x, y, name),  
 side\_a(sideLength) {}  
  
void Square::display() {  
 cout << setprecision(2) << fixed;  
 cout << "Square Name: " << shapeName << endl;  
 cout << "X-coordinate: " << origin.getx() << endl;  
 cout << "Y-coordinate: " << origin.gety() << endl;  
 cout << "Side a: " << side\_a << endl;  
 cout << "Area: " << area() << endl;  
 cout << "Perimeter: " << perimeter() << endl;  
}  
  
const double Square::perimeter() const{  
 return 4 \* side\_a;  
}  
  
const double Square::area()const {  
 return side\_a \* side\_a;  
}  
  
//double Square::distance(Square &other) {  
// Point p1(origin.getx() + side\_a / 2, origin.gety() + side\_a / 2);  
// Point p2(other.origin.getx() + other.side\_a / 2, other.origin.gety() + other.side\_a / 2);  
// return p1.distance(p2);  
//}  
//  
//double Square::distance(Square &the\_shape, Square &other) {  
// Point p1(the\_shape.origin.getx() + the\_shape.side\_a / 2, the\_shape.origin.gety() + the\_shape.side\_a / 2);  
// Point p2(other.origin.getx() + other.side\_a / 2, other.origin.gety() + other.side\_a / 2);  
// return p1.distance(p2);  
//}

**Exercise B: Templates in C++ (10 marks)**

The first element of vector x contains: 999

Testing an <int> Vector:

Testing sort

-77 0

88 1

999 2

Testing Postfix ++:

0

1

2

Testing Prefix --:

2

1

0

Testing Prefix ++:

1

2

0

Testing Postfix --

0

2

1

Testing a <Mystring> Vector:

Testing sort

All 0

Bar 1

Foo 2

Testing Postfix ++:

0

1

2

Testing Prefix --:

2

1

0

Testing Prefix ++:

1

2

0

Testing Postfix --

0

2

1

Testing a <char \*> Vector:

Testing sort

Apple 0

Orange 1

Pear 2

Testing Postfix ++:

0

1

2

Prgram Terminated Successfully.

Process finished with exit code 0

/\*  
\* File Name: iterator.cpp  
\* Assignment: Lab 3 Exercise B  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/

#include <iostream>  
#include "cstring"  
#include <assert.h>  
#include "mystring2.h"  
  
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 &x);  
  
 int& 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.  
  
 int 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.  
  
 int& 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.  
  
 int 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\*();  
  
// template <class Mystring>  
// Mystring 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  
public:  
};  
  
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>  
Vector<T>::VectIter::VectIter(Vector &x) {  
 v = &x;  
 index = 0;  
}  
  
template<class T>  
int &Vector<T>::VectIter::operator++() {  
 index++;  
 if (index >= v->size) {  
 index = 0;  
 }  
 return index;  
}  
  
template<class T>  
int Vector<T>::VectIter::operator++(int) {  
 int temp = index;  
 index++;  
 if (index >= v->size) {  
 index = 0;  
 }  
 return temp;  
}  
  
template<class T>  
int Vector<T>::VectIter::operator--(int) {  
 int temp = index;  
 index--;  
 if (index < 0) {  
 index = v->size - 1;  
 }  
 return temp;  
}  
  
template<class T>  
int &Vector<T>::VectIter::operator--() {  
 index--;  
 if (index < 0) {  
 index = v->size - 1;  
 }  
 return index;  
}  
  
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 << "\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  
  
#if 1  
 cout << "\nTesting an <int> Vector: " << endl;  
  
 cout << "\n\nTesting sort";  
 x.ascending\_sort();  
  
 for (int i = 0; i < 3; i++)  
 cout << endl << \*iter << " " << iter++;  
  
 cout << "\n\nTesting Postfix ++:";  
 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;  
  
 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 << " " << iters++;  
  
 cout << "\n\nTesting Postfix ++:";  
 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";  
 z.ascending\_sort();  
 for (int i = 0; i < 3; i++)  
 cout << endl << \*iterchar << " " << iterchar++;  
  
 cout << "\n\nTesting Postfix ++:";  
 for (int i = 0; i < 3; i++)  
 cout << endl << iterchar++;  
  
#endif  
 cout << "\nPrgram Terminated Successfully." << endl;  
  
 return 0;  
}

**Exercise C (20 marks)**

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)-1111-123334

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:: (403)-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.

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.

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.

Test assignment operator: key should be 8001

8001 C++ is a powerful language for engineers but it's not easy.

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.

Process finished with exit code 0

/\*  
\* File Name: mainLab3ExC.cpp  
\* Assignment: Lab 3 Exercise C  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/

#include <cassert>  
#include <iostream>  
#include "lookupTable.h"  
#include <cstring>  
  
using namespace std;  
  
template<class T, class K>  
void print(LookupTable<T, K> &lt);  
  
template<class T, class K>  
void try\_to\_find(LookupTable<T, K> &lt, int 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 T, class K>  
void print(LookupTable<T, K> &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 T, class K>  
void try\_to\_find(LookupTable<T, K> &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";  
}  
  
void test\_Customer()  
//creating a lookup table for customer objects.  
{  
 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<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<int, Customer> 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()  
  
// creating lookuptable for Mystring objects  
{  
 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<int, Mystring> 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()  
  
//creating look table of integers  
  
{  
 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<int, int> 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";  
  
}

/\*  
\* File Name: lookupTable.h  
\* Assignment: Lab 3 Exercise C  
\* Lab Section: B01  
\* Completed by: Kaumil Patel  
\* Submission Date: Oct 14, 2021  
\*/

#ifndef LookupTable\_H  
#define LookupTable\_H  
  
#include <iostream>  
  
using namespace std;  
  
// class LookupTable<T, K>: 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<T, K> 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<T, K> a new struct type called Pair  
// is introduced which represents a key/data pair.  
  
  
template<class T, class K>  
struct Pair {  
 Pair(T keyA, K datumA) : key(keyA), datum(datumA) {  
 }  
 T key;  
 K datum;  
};  
  
template<typename T, typename K> class LookupTable;  
template<typename T, typename K>  
ostream& operator<< (ostream &, const LookupTable<T,K>& lt);  
  
template<class T, class K>  
class LT\_Node {  
 friend class LookupTable<T, K>;  
  
private:  
 Pair<T, K> pairM;  
 LT\_Node \*nextM;  
  
 // This ctor should be convenient in insert and copy operations.  
 LT\_Node(const Pair<T, K> &pairA, LT\_Node \*nextA);  
};  
  
template<class T, class K>  
class LookupTable {  
public:  
 // Nested class  
 class Iterator {  
 friend class LookupTable;  
  
 LookupTable \*LT;  
// LT\_Node\* cursor;  
 public:  
 Iterator() : LT(0) {}  
  
 Iterator(LookupTable &x) : LT(&x) {}  
  
 const K &operator\*();  
  
 const K &operator++();  
  
 const K &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 &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 T &cursor\_key() const;  
 // REQUIRES: cursor\_ok()  
 // PROMISES: Returns key of key/datum pair to which cursor is attached.  
  
 const K & cursor\_datum() const;  
 // REQUIRES: cursor\_ok()  
 // PROMISES: Returns datum of key/datum pair to which cursor is attached.  
  
 void insert(const Pair<T, K> &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 T &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 T &keyA);  
 // PROMISES:  
 // If keyA matches a key in the table, the cursor is attached  
 // to the corresponding key/datum pair.  
 // If keyA does not match an existing key, the cursor is put in  
 // the off-list state.  
  
 void go\_to\_first();  
 // PROMISES: If size() > 0, cursor is moved to the first key/datum pair  
 // in the table.  
  
 void step\_fwd();  
 // REQUIRES: cursor\_ok()  
 // PROMISES:  
 // If cursor is at the last key/datum pair in the list, cursor  
 // goes to the off-list state.  
 // Otherwise the cursor moves forward from one pair to the next.  
  
 void make\_empty();  
 // PROMISES: size() == 0.  
  
  
 friend ostream &operator<< <T, K>(ostream &os, const LookupTable<T, K> &lt);  
  
private:  
 int sizeM;  
 LT\_Node<T, K> \*headM;  
 LT\_Node<T, K> \*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.  
};  
  
  
  
template<class T, class K>  
LookupTable<T, K> &LookupTable<T, K>::begin() {  
 cursorM = headM;  
 return \*this;  
}  
  
template<class T, class K>  
LT\_Node<T, K>::LT\_Node(const Pair<T, K> &pairA, LT\_Node<T, K> \*nextA)  
 : pairM(pairA), nextM(nextA) {  
}  
  
template<class T, class K>  
LookupTable<T, K>::LookupTable()  
 : sizeM(0), headM(0), cursorM(0) {  
}  
  
template<class T, class K>  
LookupTable<T, K>::LookupTable(const LookupTable &source) {  
 copy(source);  
}  
  
template<class T, class K>  
LookupTable<T, K> &LookupTable<T, K>::operator=(const LookupTable &rhs) {  
 if (this != &rhs) {  
 destroy();  
 copy(rhs);  
 }  
 return \*this;  
}  
  
template<class T, class K>  
LookupTable<T, K>::~LookupTable() {  
 destroy();  
}  
  
template<class T, class K>  
int LookupTable<T, K>::size() const {  
 return sizeM;  
}  
  
template<class T, class K>  
int LookupTable<T, K>::cursor\_ok() const {  
 return cursorM != 0;  
}  
  
template<class T, class K>  
const T &LookupTable<T, K>::cursor\_key() const {  
 assert(cursor\_ok());  
 return cursorM->pairM.key;  
}  
  
template<class T, class K>  
const K &LookupTable<T, K>::cursor\_datum() const {  
 assert(cursor\_ok());  
 return cursorM->pairM.datum;  
}  
  
template<class T, class K>  
void LookupTable<T, K>::insert(const Pair<T, K> &pairA) {  
 // Add new node at head?  
 if (headM == 0 || pairA.key < headM->pairM.key) {  
 headM = new LT\_Node<T, K>(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<T, K> \*before = headM;  
 LT\_Node<T, K> \*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<T, K>(pairA, before->nextM);  
 sizeM++;  
 }  
 }  
}  
  
template<class T, class K>  
void LookupTable<T, K>::remove(const T &keyA) {  
  
 if (headM == 0 || keyA < headM->pairM.key)  
 return;  
  
 LT\_Node<T, K> \*doomed\_node = 0;  
 if (keyA == headM->pairM.key) {  
 doomed\_node = headM;  
 headM = headM->nextM;  
 sizeM--;  
 } else {  
 LT\_Node<T, K> \*before = headM;  
 LT\_Node<T, K> \*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<class T, class K>  
void LookupTable<T, K>::find(const T &keyA) {  
 LT\_Node<T, K> \*ptr = headM;  
 while (ptr != NULL && ptr->pairM.key != keyA) {  
 ptr = ptr->nextM;  
 }  
  
 cursorM = ptr;  
}  
  
template<class T, class K>  
void LookupTable<T, K>::go\_to\_first() {  
 cursorM = headM;  
}  
  
template<class T, class K>  
void LookupTable<T, K>::step\_fwd() {  
 assert(cursor\_ok());  
 cursorM = cursorM->nextM;  
}  
  
template<class T, class K>  
void LookupTable<T, K>::make\_empty() {  
 destroy();  
 sizeM = 0;  
 cursorM = 0;  
}  
  
template<class T, class K>  
void LookupTable<T, K>::destroy() {  
  
 LT\_Node<T, K> \*ptr = headM;  
 while (ptr != NULL) {  
 headM = headM->nextM;  
 delete ptr;  
 ptr = headM;  
  
 }  
 cursorM = NULL;  
 sizeM = 0;  
}  
  
template<class T, class K>  
void LookupTable<T, K>::copy(const LookupTable &source) {  
  
 headM = 0;  
 cursorM = 0;  
  
 if (source.headM == 0)  
 return;  
  
 for (LT\_Node<T, K> \*p = source.headM; p != 0; p = p->nextM) {  
 insert(Pair<T, K>(p->pairM.key, p->pairM.datum));  
 if (source.cursorM == p)  
 find(p->pairM.key);  
 }  
  
}  
  
template <class T, class K>  
ostream &operator<<(ostream &os, const LookupTable<T, K> &lt) {  
 if (lt.cursor\_ok())  
 os << lt.cursor\_key() << " " << lt.cursor\_datum();  
 else  
 os << "Not Found.";  
  
 return os;  
}  
  
template<class T, class K>  
const K &LookupTable<T, K>::Iterator::operator\*() {  
 assert(LT->cursor\_ok());  
 return LT->cursor\_datum();  
}  
  
template<class T, class K>  
const K &LookupTable<T, K>::Iterator::operator++() {  
 assert(LT->cursor\_ok());  
 const K &x = LT->cursor\_datum();  
 LT->step\_fwd();  
 return x;  
}  
  
template<class T, class K>  
const K &LookupTable<T, K>::Iterator::operator++(int) {  
 assert(LT->cursor\_ok());  
  
 LT->step\_fwd();  
 return LT->cursor\_datum();  
}  
  
template<class T, class K>  
int LookupTable<T, K>::Iterator::operator!() {  
 return (LT->cursor\_ok());  
}  
  
#endif