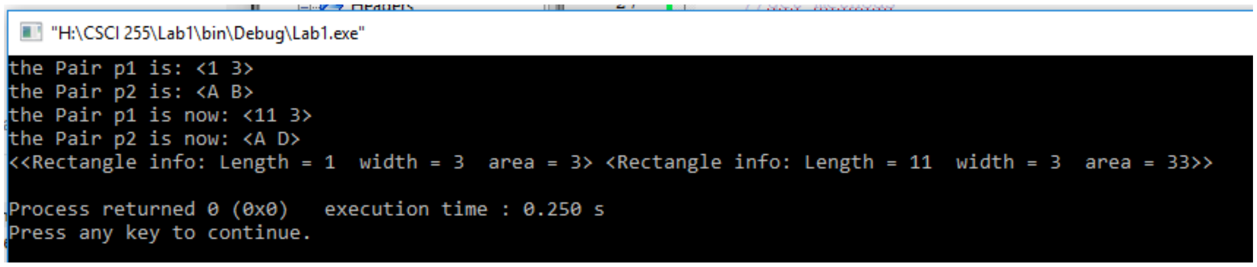
Darwin Groskleg
200903596



"H:\CSCI 255\Lab1\bin\Debug\Lab1.exe"

```
the Pair p1 is: <1 3>
the Pair p2 is: <A B>
the Pair p1 is now: <11 3>
the Pair p2 is now: <A D>
<<Rectangle info: Length = 1  width = 3  area = 3> <Rectangle info: Length = 11  width = 3  area = 33>>

Process returned 0 (0x0)   execution time : 0.250 s
Press any key to continue.
```

**main.cpp**

```cpp
#include <iostream>
#include "pair.h"
#include "rectangle.h"

using namespace std;

/* Creates 2 Pair objects and assigns values to each,
 * displays their values on the screen,
 * mutates their values and displays it again.
 */
int main() {
    Pair<int>  pair1(1, 3);
    Pair<char> pair2('A', 'B');

    cout << "the Pair pair1 is: ";
    pair1.display();

    cout << "the Pair pair2 is: ";
    pair2.display();

    pair1.setElement(1, 11);
    pair2.setElement(2, 'D');

    cout << "the Pair pair1 is now: ";
    pair1.display();

    cout << "the Pair pair2 is now: ";
    pair2.display();


    Pair<Rectangle> two_rectangles{ Rectangle(1, 3), Rectangle(11, 3) };
    two_rectangles.display();

    return 0;
}
```

**pair.h**

```
#ifndef PAIR_H_INCLUDED
#define PAIR_H_INCLUDED

#include <iostream>

template<class T>
class Pair {
    T first_;
    T second_;

public:
    // null object
    Pair() :
        first_(nullptr),
        second_(nullptr)
        {};

    Pair(T first, T second) :
        first_(first),
        second_(second)
        {};

    void display();

    void setElement(int element, T value);
};

template<class T>
void Pair<T>::display() {
    std::cout << '<' << first_ << ' ' << second_ << ">\n";
}

template<class T>
void Pair<T>::setElement(int element, T value) {
    if (element == 1)
        first_ = value;
    else if(element == 2)
        second_ = value;
}

template<class T>
void Pair<T>::display() {
    std::cout << '<' << first_ << ' ' << second_ << ">\n";
}

template<class T>
void Pair<T>::setElement(int element, T value) {
    if (element == 1)
        first_ = value;
    else if(element == 2)
        second_ = value;
}
#endif // PAIR_H_INCLUDED
```

**rectangle.h (changes: overloaded the << operator)**

```
/* *******************************************************
 * purpose: demonstrate class concept in C++
 */
#ifndef rectangle_h
#define rectangle_h
#include <iostream>
#include <iomanip>

using namespace std;

//define the Rectangle class
class Rectangle
{
    //data members, the length and widther of Retangle
  private:
    int length;
    int width;

  public:
    Rectangle();
    Rectangle (int l, int w);  //constructor
    //get methods
    int getLength();
    int getWidth();
    //set methods
    void setLength(int l);
    void setWidth(int w);
    //method to compute the area of the Rectangle
    int computeArea(); //note that no need for arguement,
                       //length and width are already store in the object

    void display(); //display the Rectangle object

    friend ostream& operator<<(ostream& os, Rectangle& rect);
};

ostream& operator<<(ostream& os, Rectangle& rect)
{
    os << "<Rectangle info: Length = "  << rect.getLength()
    << "  width = " << rect.getWidth()
    << "  area = " << rect.computeArea() << '>';
    return os;
}

//implementation of methods
//constructor with no arguments
Rectangle::Rectangle ()
{
    length = 0;
    width = 0;
}

//constructor with two arguements
```

```
Rectangle::Rectangle (int l, int w)
{
    setLength(l);
    setWidth(w);
}

//get the length of the Rectangle
int Rectangle::getLength()
{
    return length;
}

//get the width of the Rectangle
int Rectangle::getWidth()
{
    return width;
}

//set the length of the Rectangle
void Rectangle::setLength(int l)
{
    length = l;
}

//set the width of the Rectangle
void Rectangle::setWidth(int w)
{
    width = w;
}


//method to compute the area of the Rectangle
int Rectangle::computeArea()
{
    return length * width;
}

//display the info of the Rectangle object
void Rectangle::display()
{
    cout << "Rectangle info: Length = "  << length
    << "  width = " << width
    << "  area = " << computeArea() << endl;
}
#endif
```