CSCI 140 PA 7 Submission

Due Date: 4/15/23

Name: Ali Mortada

Exercise 1 – **zyBook 7.24 LAB: Winning team** -- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score

Exercise 2 – **zyBook 7.26 LAB: Artwork label** -- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score

Exercise 3 – **zyBook 7.28 LAB*: Program: Online shopping cart (Part 2)** -- check if completely done in zyBook ____ ; otherwise, discuss issues below
Include a screenshot of current status/score

LAB ACTIVITY | 7.28.1: LAB*: Program: Online shopping cart (Part 2) | 11 / 31

Current file: main.cpp | Load default template...

```
#include <iostream>
#include <iomanip>
#include <string>
#include <vector>
using namespace std;

#include "ShoppingCart.h"

void PrintMenu() {
   cout << "MENU" << endl;
   cout << "a - Add item to cart" << endl;
   cout << "d - Remove item from cart" << endl;
   cout << "c - Change item quantity" << endl;
   cout << "i - Output items' descriptions" << endl;
   cout << "o - Output shopping cart" << endl;
   cout << "q - Quit" << endl;
}
```

Was unable to complete all test cases. Code had some kind of infinite loop but I wasn't able to figure out where it was.

Exercise 5 – **Height Class Version 2** -- check if completely done __X__ ; otherwise, discuss issues below

Source code below:

Driver program:

```
/*
    Program: Height Class V2
    Author: Ali Mortada
    Class: CSCI 140
    Date: 4/15/23
    Description: Driver program for Height class. Stores height as feet
and inches in a class.
    Can also print stored height and adjust height by up to 12 inches.
    I certify that the code below is my own work.
    Exception(s): N/A
*/

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

int main() {

    cout << "Author: Ali Mortada" << endl << endl;

    // Create Height objects
    Height h3(5, 8);           // 5 feet, 8 inches, output: 5' 8"
    Height h4(-1, 5);          // 0 feet, 5 inches, output: 0' 5" (default
feet used)
    Height h5(6, 15);          // 6 feet, 0 inches, output: 6' 0" (default
inches used)

    // Print height h3
    cout << "h3: ";
    h3.print();                // h3: 5' 8"
    cout << endl;

    // Print height h4
    cout << "h4: ";
    h4.print();                // h4: 0' 5"
    cout << endl;

    // Print height h5
```

```
    cout << "h5: ";
    h5.print();                // h5: 6' 0"
    cout << endl;

    // Perform various operations
    h3.setFeet(-2);            // 5 feet, 8 inches (feet stay the same)
    h3.setInches(10);          // 5 feet, 10 inches
    cout << "feet: " << h3.getFeet() << ", inches: " << h3.getInches() <<
endl;  // 5' 10"

    h4.setFeet(6);             // 6 feet, 5 inches
    h4.setInches(12);          // 6 feet, 5 inches (inches stay the same)
    cout << "feet: " << h4.getFeet() << ", inches: " << h4.getInches() <<
endl;  // 6' 5"

    h5.setInches(10);          // 6 feet, 10 inches
    h5.adjust();               // 6 feet, 11 inches
    h5.adjust();               // 7 feet, 0 inches
    h5.print();                // 7' 0"

    cout << endl;

    Height h6(0, 0);           // 0 feet, 0 inches

    // Print height
    h6.print();                // h6: 0' 0"

    // Test new adjust and totalInches operations
    h6.adjust(12);             // 1 foot, 0 inches
    h6.adjust(-3);             // 0 feet, 9 inches
    h6.adjust();               // 0 feet, 10 inches
    h6.adjust(10);             // 1 foot, 8 inches
    cout << "Total inches: " << h6.totalInches() << endl;    // 20 inches

    // Test overloaded == operator
    if (h3 == h6) {
        cout << "h3 is the same as h6" << endl;
    } else {
        cout << "h3 is not the same as h6" << endl;
    }
    // Should output "h3 is not the same as h6"

    return 0;
}
```

Class declaration:

```
#ifndef HEIGHT_H
#define HEIGHT_H

#include <iostream>
using namespace std;

class Height {
    public:
        Height(int userFeet, int userInches);
        bool operator==(const Height &r) const;

        void setFeet(int userFeet);
        void setInches(int userInches);

        int getFeet() const;
        int getInches() const;
        int totalInches() const;

        void print() const;
        void adjust();
        void adjust(int userInches);

    private:
        int feet;
        int inches;
};

#endif
```

Class definition:

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

// Default constructor
Height::Height(int userFeet, int userInches) {
    // Set feet to input if >= 0, if not then default to 0
    if (userFeet >= 0) {
        feet = userFeet;
    } else {
        feet = 0;
    }
    // Set inches to input if between 0 and 11, if not then default to 0
    if (userInches >= 0 && userInches <= 11) {
        inches = userInches;
    } else {
        inches = 0;
    }
}

// Operator overload for ==, allows two heights to be compared
bool Height::operator==(const Height &r) const {
    if (feet == r.feet && inches == r.inches) {
        return true;
    }
    return false;
}

// Mutator for feet data member
void Height::setFeet(int userFeet) {
    if (userFeet >= 0)
        feet = userFeet;
}

// Mutator for inches data member
void Height::setInches(int userInches) {
    if (userInches >= 0 && userInches <= 11)
        inches = userInches;
}

// Accessor for feet data member
int Height::getFeet() const {
    return feet;
```

```
}

// Accessor for inches data member
int Height::getInches() const {
    return inches;
}

// Returns total inches
int Height::totalInches() const {
    return (feet * 12) + inches;
}

// Print height in ' " format
void Height::print() const {
    cout << feet << "' " << inches << "\"" << endl;
}

// Increments inch by 1 and, if needed, feet by 1
void Height::adjust() {
    if (inches != 11) {
        inches++;
    } else {
        inches = 0;
        feet++;
    }
}

// Adjusts height by a given amount of inches
void Height::adjust(int userInches) {
    inches += userInches;
    // If inches overflow, carry a foot up or down
    if (inches >= 12) {
        inches -= 12;
        feet++;
    }
    if (inches < 0) {
        inches += 12;
        feet--;
    }
}
```

Input/output below:

```
PS C:\Users\Ali\Documents\Mt. SAC\CSCI 140\CSCI-140\PA 7> g++ HeightApp.cpp Height.cpp -o HeightApp
PS C:\Users\Ali\Documents\Mt. SAC\CSCI 140\CSCI-140\PA 7> ./HeightApp
Author: Ali Mortada

h3: 5' 8"

h4: 0' 5"

h5: 6' 0"

feet: 5, inches: 10
feet: 6, inches: 5
7' 0"

0' 0"
Total inches: 20
h3 is not the same as h6
```

Answer for Question 1

Yes, it is possible to have a private int variable to represent the total inches in the height without changing any public member functions or changing the way the application works. This variable would just have to be initialized to the correct value when the object is created and modified each time the object's height is modified, and the public member function totalInches would simply return this variable.

Answer for Question 2

One good reason for overloading operators in a class is that it allows you to compare objects using relational operators without having to compare every single data member, making your code more modular, compact, and readable. Another reason is that it allows you to do arithmetic with all data members of an object without having to always use the class's getter functions.

Extra Credit – provide if applicable.

Source code:

```
/*
    Program: Height Class V2
    Author: Ali Mortada
    Class: CSCI 140
    Date: 4/15/23
    Description: Driver program for Height class. Stores height as feet
and inches in a class.
    Can also print stored height and adjust height by up to 12 inches.
Operators overloaded include
    ==, <, and <<, allowing comparison of two objects and stream
insertion.
    I certify that the code below is my own work.
    Exception(s): N/A
*/

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

int main() {

    cout << "Author: Ali Mortada" << endl << endl;

    // Create Height objects
    Height h3(5, 8);           // 5 feet, 8 inches, output: 5' 8"
    Height h4(-1, 5);          // 0 feet, 5 inches, output: 0' 5" (default
feet used)
    Height h5(6, 15);          // 6 feet, 0 inches, output: 6' 0" (default
inches used)

    // Print height h3
    cout << "h3: ";
    h3.print();                // h3: 5' 8"
    cout << endl;

    // Print height h4
    cout << "h4: ";
    h4.print();                // h4: 0' 5"
    cout << endl;

    // Print height h5
    cout << "h5: ";
    h5.print();                // h5: 6' 0"
```

```
    cout << endl;

    // Perform various operations
    h3.setFeet(-2);            // 5 feet, 8 inches (feet stay the same)
    h3.setInches(10);          // 5 feet, 10 inches
    cout << "feet: " << h3.getFeet() << ", inches: " << h3.getInches() <<
endl;  // 5' 10"

    h4.setFeet(6);             // 6 feet, 5 inches
    h4.setInches(12);          // 6 feet, 5 inches (inches stay the same)
    cout << "feet: " << h4.getFeet() << ", inches: " << h4.getInches() <<
endl;  // 6' 5"

    h5.setInches(10);          // 6 feet, 10 inches
    h5.adjust();               // 6 feet, 11 inches
    h5.adjust();               // 7 feet, 0 inches
    h5.print();                // 7' 0"

    cout << endl;

    Height h6(0, 0);           // 0 feet, 0 inches

    // Print height
    h6.print();                // h6: 0' 0"

    // Test new adjust and totalInches operations
    h6.adjust(12);             // 1 foot, 0 inches
    h6.adjust(-3);             // 0 feet, 9 inches
    h6.adjust();               // 0 feet, 10 inches
    h6.adjust(10);             // 1 foot, 8 inches
    cout << "Total inches: " << h6.totalInches() << endl;    // 20 inches

    // Test overloaded == operator
    if (h3 == h6) {
        cout << "h3 is the same as h6" << endl;
    } else {
        cout << "h3 is not the same as h6" << endl;
    }
    // Should output "h3 is not the same as h6"

    // h3 is 5' 10" and h6 is 1' 8"
    if (h3 < h6) {
        cout << "h3 is less than h6" << endl;
    } else if (h3 == h6){
        cout << "h3 is equal to h6" << endl;
```

```
    } else {
        cout << "h3 is greater than h6" << endl;
    }
    // Should output: "h3 is greater than h6"

    cout << "h3: " << h3 << endl;    // h3: 5' 10" (70 inches)
    cout << "h6: " << h6 << endl;    // h6: 1' 8" (20 inches)

    return 0;
}
```

Class declaration:

```
#ifndef HEIGHT_H
#define HEIGHT_H

#include <iostream>
using namespace std;

class Height {
    public:
        Height(int userFeet, int userInches);
        bool operator==(const Height &r) const;
        bool operator<(const Height &r) const;
        friend ostream& operator<<(ostream& os, const Height &r);

        void setFeet(int userFeet);
        void setInches(int userInches);

        int getFeet() const;
        int getInches() const;
        int totalInches() const;

        void print() const;
        void adjust();
        void adjust(int userInches);

    private:
        int feet;
        int inches;
};

#endif
```

Class definition:

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

// Default constructor
Height::Height(int userFeet, int userInches) {
    // Set feet to input if >= 0, if not then default to 0
    if (userFeet >= 0) {
        feet = userFeet;
    } else {
        feet = 0;
    }
    // Set inches to input if between 0 and 11, if not then default to 0
    if (userInches >= 0 && userInches <= 11) {
        inches = userInches;
    } else {
        inches = 0;
    }
}

// Operator overload for ==, allows two heights to be compared
bool Height::operator==(const Height &r) const {
    if (feet == r.feet && inches == r.inches) {
        return true;
    }
    return false;
}

// Operator overload for <, allows two heights to be compared
bool Height::operator<(const Height &r) const {
    if (feet < r.feet && inches < r.inches) {
        return true;
    }
    return false;
}

// Operator overload for <<, allows printing for heights
ostream& operator<<(ostream& os, const Height &r) {
    os << r.feet << "' " << r.inches << "\" (" << r.totalInches() << "
inches)" << endl;
    return os;
}
```

```cpp
// Mutator for feet data member
void Height::setFeet(int userFeet) {
    if (userFeet >= 0)
        feet = userFeet;
}

// Mutator for inches data member
void Height::setInches(int userInches) {
    if (userInches >= 0 && userInches <= 11)
        inches = userInches;
}

// Accessor for feet data member
int Height::getFeet() const {
    return feet;
}

// Accessor for inches data member
int Height::getInches() const {
    return inches;
}

// Returns total inches
int Height::totalInches() const {
    return (feet * 12) + inches;
}

// Print height in ' " format
void Height::print() const {
    cout << feet << "' " << inches << "\"" << endl;
}

// Increments inch by 1 and, if needed, feet by 1
void Height::adjust() {
    if (inches != 11) {
        inches++;
    } else {
        inches = 0;
        feet++;
    }
}

// Adjusts height by a given amount of inches
void Height::adjust(int userInches) {
    inches += userInches;
```

```
    // If inches overflow, carry a foot up or down
    if (inches >= 12) {
        inches -= 12;
        feet++;
    }
    if (inches < 0) {
        inches += 12;
        feet--;
    }
}
```

Input/output:

```
PS C:\Users\Ali\Documents\Mt. SAC\CSCI 140\CSCI-140\PA 7> g++ HeightApp.cpp Height.cpp -o HeightApp
PS C:\Users\Ali\Documents\Mt. SAC\CSCI 140\CSCI-140\PA 7> ./HeightApp
Author: Ali Mortada

h3: 5' 8"

h4: 0' 5"

h5: 6' 0"

feet: 5, inches: 10
feet: 6, inches: 5
7' 0"

0' 0"
Total inches: 20
h3 is not the same as h6
h3 is greater than h6
h3: 5' 10" (70 inches)

h6: 1' 8" (20 inches)
```