**LESSON SET 3**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expressions, Input, Output and Data Type**

**Conversions**

**OBJECTIVES FOR STUDENT**

**Lesson 3A:**

1. To work with input statements

2. To learn input and formatted output statements

3. To work with constants and mathematical functions

**Lesson 3B:**

4. To learn data type conversions (coercion and casting)

These lessons introduce a variety of several key concepts in programming and

are thus more comprehensive in content than most.

**ASSUMPTIONS**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lesson 3A:**

1. Students have a good understanding of the basic data types in C++

2. Students know the output, input and assignment statements in C++

**Lesson 3B:**

1. Students know the difference between constants and variables

2. Students know the concept of pre-defined functions in C++

3. Students know some basic concepts of file manipulation

**PRE-LAB WRITING ASSIGNMENT SOLUTIONS**

1. –20

2. 2\*x+pow(3,4)

3. coercion

4. casting

5. #include <iostream>

6. whitespace

7. insertion

8. iomanip

9. causes a carriage return (end of line)

**LAB ASSIGNMENTS**

As always, instructors should pick labs best suited for their particular class. All three

labs of Lesson 3A cover the basics of the chapter and thus should be completed.

Lab 3.5 is the student generated code component.

**Lesson 3A:**

Lab 3.1: Working with the cin statement

Lab 3.2: Formatting output

Lab 3.3: Arithmetic operations and math functions

**Lesson 3B:**

Lab 3.4: Working with type casting

Lab 3.5: Student generated code assignments

**LESSON 3A**

**LAB 3.1: Working with the cin Statement**

Lab 3.1 introduces the cin statement and shows the importance of prompts for

interactive input.

Students may be confused about when to use the << and >> operators. The difference

should be reinforced.

Students may have to experiment for awhile to realize that at times a cout

statement may need to begin with cout << endl <<.... This is sometimes true when a cout statement comes right after a cin statement.

A solution is found in billKEY1.cpp in the instructor’s folder for Lesson Set 3.

**Possible Answers to the Exercise Questions**

*Exercise 2:*

1. Rerun the program with the same data given in Exercise 1 above and record your results.

**4e+002 (or something to this effect). It puts the answer in scientific notation.**

2. What do you think the fixed attribute in the cout statement does?

**It puts the answer in standard decimal notation instead of scientific notation.**

*Exercise 3:*

1. Rerun the program with the same data given in Exercise 1 and record your results:

**$241.5600**

2. What do you think the setprecision( ) attribute in the cout statement does?

**It determines how many digits to the right of the decimal point are shown.**

*Exercise 4:*

This is an optional section that introduces the string class (used as a data type). A solution is found in billKEY2.cpp in the instructor’s folder for Lesson Set 3.

**LAB 3.2: Formatting Output**

Lab 3.2 works with formatted output and follows very closely the material learned

in Lab 3.1. Thus it is important that 3.1 and 3.2 are done in sequence.

A solution is found in tabledataKey.cpp in the instructor’s folder for Lesson Set 3.

**LAB 3.3: Arithmetic Operations and Math Functions**

This lab has the students convert the Pythagorean formula into a C++ expression.

It requires the use of the pow and sqrt math functions.

A solution for Exercise 1 is found in righttrigKey1.cpp in the instructor’s folder for Lesson Set 3.

In Exercise 2 of this lab the students need to alter the program by including the directive <iomanip>. They also need to include the statement:

**cout << fixed << showpoint << setprecision(2);**

just before printing the length of the hypotenuse.

A solution for Exercise 2 is found in righttrigKEY2.cpp in the instructor’s folder for Lesson Set 3.

**LESSON 3B**

**Lab 3.4: Working with Type Casting**

This lab works with the classic batting average problem. Since at bats will always

be less than hits (unless we really have a superstar), the average obtained by dividing

by these two integers will always be 0. This lab demonstrates both type casting and

coercion.

A solution is found in batavgKEY.cpp in the instructor’s folder for Lesson Set 3.

**Possible Answers to the Exercise Questions**

*Exercise 1:*

Run this program and record the results.

**The batting average is 0.**

*Exercise 2:*

There is a logical error in this program centering around data types. Does changing

the data type of batavg from int to float solve the problem?

**No. It also needs type casting.**

Make that change and run the program again and record the result.

**The batting average is 0.292162.**

**Lab 3.5: Student Generated Code Assignments**

Students should be encouraged to develop an algorithm and possibly pseudocode

before actually coding the program. The algorithm for a problem very similar to

the first option (Grade average program) was introduced in Lesson Set 1.

A solution is found in gradeKEY.cpp in the instructor’s folder for Lesson Set 3.

The second option is a simple program that calculates total sales for each of

three types of furniture.

A solution is found in furnitureKEY.cpp in the instructor’s folder for Lesson Set 3.

The third option is the most involved (in terms of the math used). Given

total sales (including state and local tax) the program must find the amount of sales

without tax and then determine both the state and local tax.

A solution is found in salestaxKEY.cpp in the instructor’s folder for Lesson Set 3.

Possible solutions to all labs are given in the instructor’s folder for Lesson Set 3.