CIS 22B Lab 4 Itty Bitty Airfreight (IBA)

#### 200 Points

Topics:

Overloaded operators

Files

-----------------------------------------------------------------------------------------------------------------------------------------

**Lab 4.1**

Utilizing the code from Lab 3.2, add an overloaded operator **==** which will be used to compare

two objects to see if they are equal. For our purposes, two objects are equal if their abbreviation

and uldid are the same. You’ll need to make your overloaded operator a friend of the Cargo class.

Create a unit1 object and load it with this data:

uld – Pallet

abbreviation – PAG

uldid – PAG32597IB

aircraft - 737

weight – 3321 **Kilograms**

destination – SEA

Make a **copy** of unit1 called unit2 utilizing a copy constructor.

Create a default unit3.

Test with code like this in **main**:

if (unit1 **==** unit2)

cout << "\n unit1 is the same as unit2 \n";

else

cout << " \nunit1 is not the same as unit2 \n";

if (unit2 **==** unit3)

cout << " \nunit2 is the same as unit3 \n";

else

cout << " \nunit2 is not the same as unit3\n";

**Lab 4.2**

Using the code from lab 4.1, add the ability to read from a file.

Modify the input function:

\* Move the input function out of the Cargo class to just below the end of the Cargo class

\* At the bottom of the input function, declare a Cargo object named

**temp** using the constructor that takes the six parameters.

\* Use the Cargo output function to print the Cargo object.

\* Call the input function from the main function with no arguments. Remove the rest of the code

from main

3. **Create a file and use it for input.** This is good because you will be

using the input many times.

\* Create a file named **cardata4.txt** (or use the one provided),

containing the following three lines of data.

If you create your own, make sure to follow the format

below:

**Pallet PAG PAG45982IB 737 4978 OAK**

**Container AYF AYF23409AA 737 2209 LAS**

**Container AAA AAA89023DL 737 5932 DFW**

\* All weights are in pounds, don’t worry about kilograms. You may

comment out that code.

\* In the input function, declare an object of type ifstream named

inputFile, which we will use to read from the file.

\* At the beginning of the code for the input function, open the

file. If the open fails, send a message to stderr and exit the

program.

\* In all the reads within the input function, remove the user

prompt and read from the inputFile object, rather than reading

from the stdin object.

\* **Hint: We need to use getline when reading the strings.**

using **>>** skips leading white space before reading the data.

getline does not skip this leading whitespace. So, **before** using

getline use the following code:

while(inputFile.peek() == ' ')

inputFile.get();

**peek** looks at the next character you are about to read. If it is

a space, **get** is used to read the space character, to get it out

of the way. Your output will then be much neater.

\* Use a loop to read each line from the file. To do this use a

while loop including all the reading in the input function, as

well building and output of the Car.

**Hint: you can do this with the following while statement:**

while(inputFile.peek() != EOF) or use this form while(inputFile >> type)

The peek function will return EOF if there is no next character.

\* At the bottom of the input function, close the file.

Put an eye catcher before the beginning of each function, class, and the

global area:

// class name function name comment(if any)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*