Lab report 3

(Carter Smith)

Compiled with “make”, on linux this compiles and runs the code

You should only have to type make when in the lab_03 directory on linux

Otherwise you can do “g++ -o main ./src/*.cpp ./include/*.h”

Sorry windows users…

If all fails download wls or put the files contained in src and include into a project file VS

Include in the submission your rationale for all members you have in your class.

Littles lots and heaps are the units we’re using.

I have 3 constructors, a default which sets everything to 0, one to set all the values to whatever integer is entered for each arg, one for just littles.

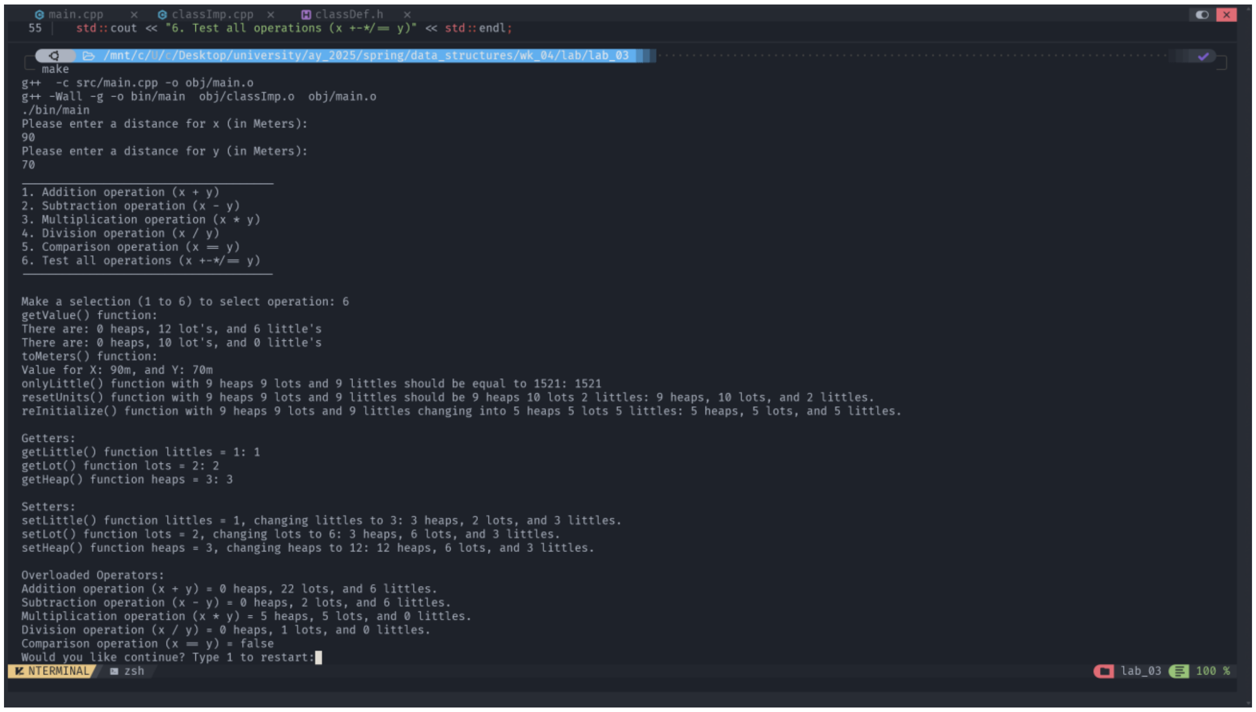
I have 5 functions, getValue() prints the values of heaps lots and littles and was required , onlyLittle() changes all units into littles, toMeters() does the same thing as onlyLittle() but I wanted to keep it streamlined inside my implementation file, reInitialization() used as a way to set a unit as another I used it to set the value for x as another object so when I did my overload operator i could print the rest of the stuff out downstream, resetUnits() this is used to change the units back into their predefined sizes.

Getters and setters, this was a requirement.

Include in the lab report any changes you needed to make to the class declaration (.h file).

The 6 overload operators and the 4 created functions toMeters, onlyLittle, toMeters, reInitialization, resetUnits.

Include in the lab report a screen shot(s) of the output of a test of all operator overload functions.



```
main.cpp  ×  classImp.cpp  ×  classDef.h  ×
    std::cout << "6. Test all operations (x +-*/== y)" << std::endl;

/mnt/c/U/c/Desktop/university/ay_2025/spring/data_structures/wk_04/lab/lab_03
  make
g++  -c src/main.cpp -o obj/main.o
g++ -Wall -g -o bin/main  obj/classImp.o  obj/main.o
./bin/main
Please enter a distance for x (in Meters):
90
Please enter a distance for y (in Meters):
70
_____________________________________
1. Addition operation (x + y)
2. Subtraction operation (x - y)
3. Multiplication operation (x * y)
4. Division operation (x / y)
5. Comparison operation (x == y)
6. Test all operations (x +-*/== y)
_____________________________________


Make a selection (1 to 6) to select operation: 6
getValue() function:
There are: 0 heaps, 12 lot's, and 6 little's
There are: 0 heaps, 10 lot's, and 0 little's
toMeters() function:
Value for X: 90m, and Y: 70m
onlyLittle() function with 9 heaps 9 lots and 9 littles should be equal to 1521: 1521
resetUnits() function with 9 heaps 9 lots and 9 littles should be 9 heaps 10 lots 2 littles: 9 heaps, 10 lots, and 2 littles.
reInitialize() function with 9 heaps 9 lots and 9 littles changing into 5 heaps 5 lots 5 littles: 5 heaps, 5 lots, and 5 littles.

Getters:
getLittle() function littles = 1: 1
getLot() function lots = 2: 2
getHeap() function heaps = 3: 3

Setters:
setLittle() function littles = 1, changing littles to 3: 3 heaps, 2 lots, and 3 littles.
setLot() function lots = 2, changing lots to 6: 3 heaps, 6 lots, and 3 littles.
setHeap() function heaps = 3, changing heaps to 12: 12 heaps, 6 lots, and 3 littles.

Overloaded Operators:
Addition operation (x + y) = 0 heaps, 22 lots, and 6 littles.
Subtraction operation (x - y) = 0 heaps, 2 lots, and 6 littles.
Multiplication operation (x * y) = 5 heaps, 5 lots, and 0 littles.
Division operation (x / y) = 0 heaps, 1 lots, and 0 littles.
Comparison operation (x == y) = false
Would you like continue? Type 1 to restart:
NTERMINAL  zsh                                                lab_03  100 %
```

Lab Submission:

1. Write a lab report including the following information:

a. A description of the objectives/concepts explored in this assignment including why you think they are important to this course and a career in CS and/or Engineering.

Some of the concepts explored in this lab are file organization, overloaded operators, and using classes and constructors. These concepts are important to CS and for engineering because all these things allow for more levels of abstraction and whether you're in CS or in any other engineering field the things you make the end user should be able to use it without caring about how it works.

b. Why you designed the class the way you did initially, what changes you made because of each task and what considerations you consider important when designing classes.

The class was designed in the way it was initially because I'm an idiot and I made the overloaded operators return the class type. This is what intuitively made sense at the time because what good is a unit if you can't use it. Like x = 7 + 9, this is an example of an integer being set to 2 integers added together but I digress. There weren't very many changes other than writing things to accept bigger things and to get things into the right format.