README

Sp2025Lab3

Group members:

- Ansh Mendiratta (Wrote class member implementations, `unit_tests.cpp`, and `main.cpp`)
- Khoi Tran (Tested class and program implementation)
- Thanh Tin

Compilation

Run `make`. If this does not work, run `g++ --std=c++2b main.cpp heaplotlittle.hpp heaplotlittle.cpp; ./a.out`

To run tests, run `make test`. This runs everything in `unit_tests.cpp` but not the required test from the assignment.

Concepts Explored

This lab tests some convenient features of languages like C++ wherein you define normally built-in behavior for your specific structures. By its nature of being OOP, it is already made important in the industry. Beyond that, however, language features like this let you fine-tune the public API a user may have and perform any necessary sanitation as well as niceties such as printing.

The design of the class was easy to foresee from the beginning. The only real changes made to the `.hpp` file was the changing of the types of each heap, lot, and little. Initially, these were `size_t`s. When the idea of a meter was mentioned and user input, we changed these to instead use `double` as no restrictions were made on what the user could input.

Class Declaration

Additional procedures that weren't required by the assignment:

- `Measurement::to_str()`: returns a string (from a stream) that prints out the measurement's quantities line by line.
- `Measurement::rebalance()`: rebalances the quantities if each of them (minus the heaps) can be "compacted".
- `Measurement::to_littles()`: uses the instance member variable values to convert into an equivalent `littles` count. Used for `rebalance()` and the arithmetic operations.

- `Measurement::to_littles()` : uses the instance member variable values to convert into an equivalent `littles` count. Used for `rebalance()` and the arithmetic operations.

Assumptions

1. A `little` is equal to 1 meter.

Test Results

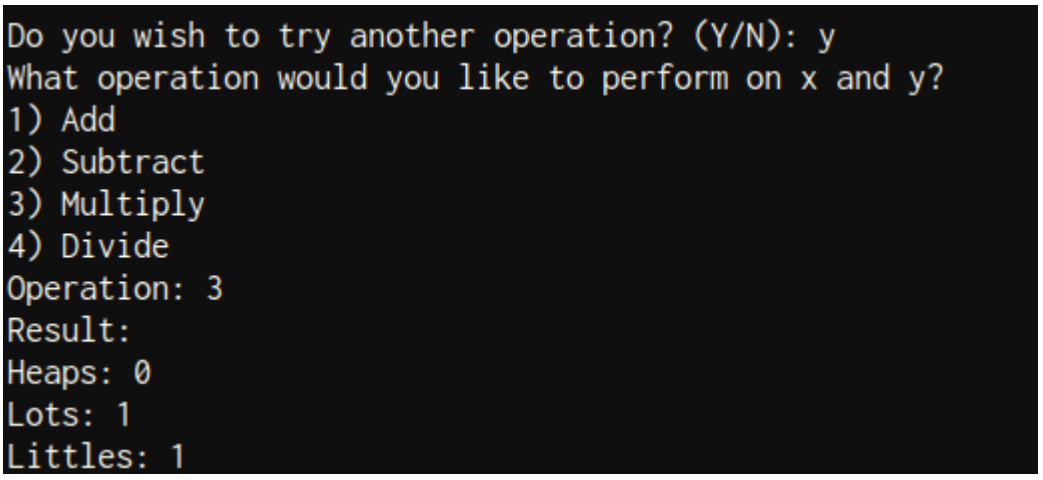
Addition:

```
Enter a distance x in meters: 2
Enter a distance y in meters: 4
What operation would you like to perform on x and y?
1) Add
2) Subtract
3) Multiply
4) Divide
Operation: 1
Result:
Heaps: 0
Lots: 0
Littles: 6
```

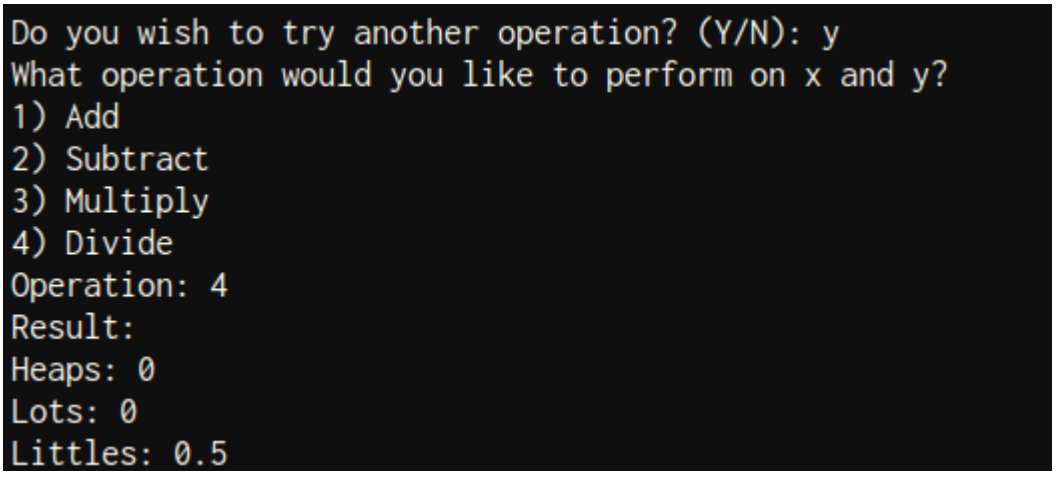
Subtraction:

```
Do you wish to try another operation? (Y/N): y
What operation would you like to perform on x and y?
1) Add
2) Subtract
3) Multiply
4) Divide
Operation: 2
Subtraction operation would result in a negative distance.
Result:
Heaps: 0
Lots: 0
Littles: 2
```

Multiplication:



Division:



The overload `operator<<` results in:

```
Heaps: ...
Lots: ...
Littles: ...
```

as visible in the screenshots.