Computer Science 381
Programming Unix in C
The College of Saint Rose
Winter Immersion 2014

Lab 8: “Object-Oriented” C

Recommended Due Date: Monday, January 13, 2014

In this lab, you will see how to develop a data structure in C using a somewhat “object-oriented” methodology, despite the lack of language features to support object orientation.

Previous Example Revisited

One of the last lab’s examples demonstrates this: the `ratio.h` and `ratio.c` files.

The structure definition and the four functions provide similar capabilities to a class in a language like Java. The fields of the structure are like instance variables of a Java class (though they would be more like `public` rather than the preferred `protected` or `private` instance variables). The `create_ratio` function plays the role of a constructor. The other functions are like the methods of a class.

Notice, however, that we have no mechanism for real data hiding. Anyone with access to a pointer to a `ratio` can access and modify the data fields. In fact, with the ability to use casts, we could treat the memory that was allocated as a `ratio` in completely unexpected (and likely incorrect) ways.

Also, all calls to functions that operate on a `ratio` must take the `ratio` as a parameter. So what would be something like

```
rat.print_ratio();
```

in an object-oriented language would be

```
print_ratio(rat);
```

here. This is what actually happens behind the scenes with real objects.

Practice Program:
Add a function `print_ratio_mixed` to `ratio.c` and a function prototype to `ratio.h` that will print a ratio as a mixed number. For example, if the `numerator` field is 8 and the `denominator` field is 3, it would print as “2 2/3” instead of “8/3”. (5 points)

A More Complex Example

Consider this example, which is an implementation of a singly-linked list structure that holds `int` values.

See Example:
`/home/cs381/examples/sll`

Study this code to make sure you know how it works.

? **Lab Question 1:**
Briefly explain why it is important to provide separate `.h` and `.c` files for an implementation of a data structure such as this. (4 points)

? **Lab Question 2:**
Draw a memory diagram showing the state of the program's memory just before it executes the `printf` statement at line 68 of `slltest.c`. Differentiate between stack variables (the contents of parameters and/or local variables of functions in execution) and heap variables (everything allocated with `malloc`). (6 points)

Programming Assignment: A C Q

Develop a queue structure and corresponding functions to operate on queues in C that hold `int` values. Include an approrpriate header file, implementation file, and a separate file with a `main` function that tests your implementation. Also include a `Makefile` that compiles your queue implementation and your testing code.

Recall that a queue is a first-in, first-out structure that should be able to be constructed, enqueue elements, dequeue elements, and answer whether it currently contains any element. You should also be able to deallocate the queue.

This program and its `Makefile` are worth 25 points.

The executable for the reference solution to this program is available on mogul in `/home/cs-381/labs/ooc`.

Submission

Please submit all required files as email attachments to *terescoj@strose.edu*. You are recommended to do so by Monday, January 13, 2014. Be sure to check that you have used the correct file names and that your submission matches all of the submission guidelines listed on the course home page.

Grading

Grading Breakdown	
`print_ratio_mixed` function	5 points
Lab questions	10 points
`int` queue correctness	15 points
`int` queue error checking	2 points
`int` queue memory management	2 points
`int` queue documentation	3 points
`int` queue style	2 points
`Makefile` for queue program	1 point