Assignment 3: Modules and Testing

For Part I of this assignment you will be writing various modules, module types, and functors, to better understand how they work. You will also use them so you have a better idea how they can help in practice.

For Part II you will write your own test suite from scratch, and configure it to use the Bisect code coverage tool which will verify you have good coverage with your tests. You will also need to document a few invariants in your tests, which we describe below.

As usual we will give two due dates f

The file structure

- Use the this zip file for your as
- We are giving you .mli files that vior. We provide a skeleton for *some* of the code in Part I. Your Part I and II answers will go in the files `assi.ml`, `assignment3/src/ring.ml`, `assignment3/src/postfix_calc.ml`, and `assignment3/tests/tests.ml`.
- Part I consists of `finite_group` s in `assignment3/tests/tests.ml`, as well as a `dune` file to run them. For Part I, you need to write some basic tests, and we will just check to make sure you have a couple. The bulk of the testing will be in Part II.
- Part II consists of `postfix_calc` and your own tests for that file, as well as making sure your overall coverage with `bisect` is good.

Coverage and Specifications for Part II

- You will need to incorporate the Bisect tool into your dune build file as was described in lecture, and use its output to improve the coverage of your test suite. Test coverage will be a component of your grade.
- Lastly you will need to write a special suite of tests which you will name "Specifications" which asserts five *different* specifications: either preconditions, postconditions, data structure invariants, or recursion invariants. These properties can be on *any* functions or data structures defined in *any* of the files from Part I or Part II. We gave several examples of such properties in lecture, for example `List.rev @@ List.rev l` is `l` for any list `l`. Such properties are useful to verify with quick-checking, but here you need to just test each property on specific cases. For each spec include a comment stating what the general property is; the test(s) will only test a few instances of the property.

Resources

Here are a few resources to keep in mind to help with this assignment.

- Make sure to review the More Modules lecture notes for Part I.
- If you feel like you need more on the subtleties of information hiding in functors, the Real World OCaml book chapter on functors may be worth looking at.
- For test coverage in part II this was covered in the Specification and Testing Lecture; these notes also contain links to OUnit and Bisect documentation.

Submission and Grading

- As usual, run a final `dune clean; dune build` and then upload `_build/default/assignment3.zip` to Gradescope. Note we will be giving you very little information in our report since you need to provide good coverage on your own and not rely on Gradescope. We try not to test anything tricky–we only want to test the functionality as it is precisely described in the assignment, and your own tests should be able to cover this.