# Project 5

**Due**: Thu, May 2, before midnight.
**No Late submissions, no extensions.**

**Important Reminder**: As per the course Academic Honesty Policy, cheating of any kind will minimally result in your letter grade for the entire course being reduced by one level.

This document first provides the aims of this project. It then lists the requirements as explicitly as possible. It then hints at how these requirements can be met. Finally, it describes how it can be submitted.

## Aims

The aims of this project are as follows:

- To give you more exposure to recursive programming, high-order functions and list comprehensions.
- To expose you to Erlang programming.

## Requirements

Implement all the functions specified in prj5_sol.erl using the Erlang compiler installed on `remote.cs`. Please consult the provided tests and LOG if the specs for a function are not entirely clear.

Specifically, all provided tests should pass:

```
 $ erl
...
1> c(prj5_sol).
{ok,prj5_sol}
2> prj5_sol:test().
  All 48 tests passed.
ok
3> q().
ok
4> $
```

## Provided Files

You should use the provided prj5-sol directory as a starting point for your project by copying it into your `i571/submit` directory. It contains the following files:

**prj5_sol.erl**
A skeleton file which contains the specifications for the functions you are required to write. Each function you are required to implement is set up to return a dummy `'TODO'` placeholder value.

**.gitignore**
Ensure `*.beam` files are not committed to git.

**.zipignore**
Ensure `*.beam` files are not submitted.

**README**
A template `README` which you should complete and submit.

The extras directory contains a sample LOG file.

# Tests

The tests use eunit. They are provided soon after the function being tested so that you can use the tests to clear up any ambiguities in the function specifications. Each set of tests are under conditional macros, making it easy to turn on or off the tests for a specific function.

# Hints

To start your project, copy over the provided prj5-sol directory to your `i571/submit` directory after creating a new branch.

The following points are worth noting:

- Details of specific syntax for different Erlang constructs are available in this Erlang Cheat Sheet.
- Note that the parameters for the `'TODO'` functions in the skeleton file have names starting with an `_` to avoid Erlang errors regarding singleton variables. The specification refer to the parameters without the leading `_`.
- The provided skeleton file is set up to export all functions automatically.
- An exported function `fn()` must be preceeded by the module name when used in the REPL, i.e. `prj5_sol:fn()`.
- Please look at the provided tests and LOG if the specs for a function are not entirely clear.
- Minor hints for each function are available in the provided prj5_sol.erl skeleton file.
- One irritating aspect of Erlang syntax is the use of `,` and `;` as separators. So for example, given:

  ```
  case X of
    { Pid, set } ->
      set_something1(),
      set_something2() ;
    { Pid, get } ->
      get_something1(),
      get_something2()
  end
  ```

  you are not allowed to have a trailing `;` after `get_something2()`.
- Move the `-if(false)` down in the **Test Control** section of the prj5_sol.erl to activate the tests for a function once you have implemented that function.
- Make sure you review the class Erlang slides before starting work on this project.

# Submission

Before submitting your project, update your README to specify the status of your project. Document any known issues.

Submit using a procedure similar to that used in your previous project (as usual, the gradescope submission link will be set up a few days later).