PLC: Homework 5 [125 points]

Due date: Wednesday, April 18th, 9pm
3 extra-credit points if you turn in by Tuesday, April 17th, 9pm

About This Homework

For this homework, you will try out internal verification in Agda. Also, you will propose your project (but this part will be posted after the April 8th weekend).

How to Turn In Your Solution

You should create a `hw5` subdirectory in your personal repo. You will copy files from subdirectories of the `hw5` directory in the course repo.

As for previous homeworks, you can check that you have submitted correctly by going to the URL for your subversion repository. Remember to use exactly the file names we are requesting (so do not change the names of these files).

Partners Allowed

You may work by yourself or with one partner (no more). See the instructions from `hw1` for details on how to submit your assignment if you work with a partner.

How To Get Help

You can post questions in the `hw5` section on Piazza.

You are also welcome to come to our office hours. See the course's Google Calendar, linked from the Resources tab of the Resources page on Piazza, for the locations and times for office hours.

1 Reading

Read Chapter 5 of Verified Functional Programming in Agda, available for free (on campus or VPN) here:

`https://dl-acm-org.proxy.lib.uiowa.edu/citation.cfm?id=2841316&CFID=852046702&CFTOKEN=22704153`

2 Vector operations [60 points]

Fill in the holes in `vector-todo.agda`. The various operations on vectors should have the same behavior as the corresponding operations (`init`, `last`, `filter`, `intersperse`, and `take`) from `Data.List` in Haskell. Each function is worth 12 points each. Filling in the hole in the type of `take`𝕍 is worth 5 points, and the rest of the code is worth 7. I found that using the function `2n-1` defined in `vector-todo.agda` to compute $2 * n - 1$ made the code for `intersperse`𝕍 quite easy to write, following the solution from the last homework (I posted that solution now, in the `hw4` directory of the course repo, so you can see it if you do an `svn update`).

3 Membership-memoized lists [40 points]

In the file `mmList.agda` you will find a definition of the type `mmList`. A value `mkMmList ss b p` of type `mmList S` consists of a list of strings `ss`, a boolean value `b`, and a proof `p` that `b` is equal to the result of calling `list-member` with `S` and `ss`. So `b` tells you whether or not `ss` contains the string `S`. The point of using a boolean value for something that you could easily compute is that you can avoid computing it! We are memoizing the result of the membership test, and using a proof to ensure that we are memoizing it correctly.

Fill in the definitions of `mmList-member` to check whether any string `s` (does not have to be `S`) is in the `mmList`'s list of strings; `_::mmList_` to add a new string to the front of an `mmList`; and `_++mmList_` to append two `mmList`s. You will also prove a property about these functions for the last problem in the file (thus performing an external verification about some of these internally verified functions).

4 Project proposal [25 points]

This will also be filled in soon.