PLC: Homework 3 [75 points]

Due date: Wednesday, February 8th, 10:30pm
3 point bonus if you turn in by Tuesday, February 7th, 10:30pm

About This Homework

The goal of this homework is to become familiar with the basics of using Agda for writing recursive programs and proofs. The problems are concerned with operations on the natural numbers ($\mathbb{N}$, which is the set containing 0, 1, 2, etc.).

The first thing you should do for this homework is copy the files you will be modifying, from the `hw/hw3` subdirectory of the class repository, to a new `hw3` subdirectory of your personal repository, similarly to `hw2`.

There are a number of extra-credit points possible on this assignment, so the maximum possible score is greater than 75 (but 75 is considered a perfect score).

Partners Allowed

For this homework, you may work by yourself or with one partner (no more). See the instructions for `hw2` for how to create the `ack.txt` and `partner.txt` files that are required if you work with a partner

All files should go in a hw3 subdirectory (which you create) of your personal repository. Please call the subdirectory exactly that. Do not call it `Homework 3` or `Hw3` or other variations, or we will penalize you 5 points.

How to Turn In Your Solution

Make sure you have done a "Subversion add" on the `hw3` directory you created in your personal repository, and on the `.agda` files you have to submit for this assignment, and then do a "Subversion commit" on your personal repository directory to submit them. The files you will be submitting are `nat-todo.agda`. You can commit as many times as you want up to the deadline. If you commit after the deadline, we will use the last existing version before the deadline. Work submitted after the deadline will not count.

How to Check Your Solution

As for `hw1`, you might want to make sure you have correctly submitted your solution via Subversion. Again, just go to the URL for your personal repository using a web browser, log in with your HawkId and password, and you can see what has been submitted.

Make sure each of the files you submit can be checked by Agda without any holes, yellow highlighting, or red highlighting. If any file does not check this way, we will penalize

you 5 points total (if multiple files do not check, we will still just penalize you 5 points). Any problems you do not solve should simply be removed from the `.agda` files, so that there are no holes (question marks) in the files.

How To Get Help

You can post questions in the `hw3` section on Piazza, or elsewhere on Piazza. See the course's Google Calendar, linked from Piazza, for the locations and times for office hours.

1 Reading

Read Chapter 3 of the book.

2 Arithmetic theorems [24 points]

In `nat-todo.agda`, you will find several theorems to prove. Each one is worth 6 points. If you do not prove a problem, you must remove it completely from the file, so that (as usual) the file will check with no holes or yellow or red highlighting.

For `prob1`, I expect you will have to write a new inductive proof without making use of existing proofs in `nat-thms.agda` in the IAL. But for the other three problems, you definitely want to make use of possibly multiple different theorem in `nat-thms.agda`. Direct proofs will be unbearably long and tedious.

3 Factorial with cutoff [25 points]

The factorial function multiplies all the numbers from n down to 1 (and is defined to be 1 if n is 0). The IAL has a definition near the end of `nat.agda`, for `factorial`.

I am supplying a file `fact.agda`, which you should copy to your personal repository and modify for this problem.

1. Fill in the definition of the function `fact`, which should behave like `factorial` (from the IAL), except that it mutiplies all the numbers from n down to some cutoff m, but not including m. So `fact` takes in two natural numbers, n and m, as input, and returns

   $$n * (n - 1) * \cdots * (m + 1)$$

   For example, `fact 6 3` should return

   $$6 * 5 * 4$$

which is 120. If the cutoff m is greater than or equal to the starting point n, then the function should just return 1. If your code is correct, the testcase `test-fact` will check without any yellow highlighting.

Hint: to test if the second number is greater than or equal to the first, you can use the $\geq$ function from `nat.agda` in the IAL. [15 points]

2. Now prove the theorem called `fact0-factorial` in `fact.agda`, which says that `fact n 0` is equal to `factorial n`. This may seem obvious, but it requires an inductive proof. [10 points]

4 Other recursive datatypes [28 points]

For this problem you will modify the file `paths.agda` from the `hw3` files. The file declares a datatype `path` for representing a path on a grid. Each path starts at a starting point (which is not mentioned explicitly in the path – we could interpret this to be the origin of the grid), and then proceeds by a sequence of moves (imagine they are of unit length) in one of the four cardinal directions `north`, `south`, `east`, and `west`. The path ends when the `end` constructor is reached.

1. Define a function `path-length` that takes in a `path` and returns the number of moves in the path (not counting `end` as a move). So `path-length (north (south (east end)))` would be 3. [5 points]
2. Define a function `flip-horizontal` that exchanges `east` and `west` moves in a path, returning a new path. [5 points]
3. Prove a theorem `double-flip` that states that applying `flip-horizontal` to a path twice results in the original path. [5 points]
4. Write a function `circles-block` which takes in a `path` and returns a $\mathbb{B}$ telling whether or not the path goes all the way around a block in the grid. For example, the path

   ```
   north (east (south (west (west (north end)))))
   ```

   goes all the way around a block, when it goes north, then east, then south, and then west. [8 points]
5. Write a function `returns` which takes a path and returns a $\mathbb{B}$ saying whether or not the path returns to the origin of the grid – that is, the position where the path began. (This is a little tricky.) [5 points]