CMPS 1600 Introduction to Computer Science II – Spring 14

4/18/14

11. Homework

Programming portion due **Friday 4/25/14** at 11:55pm on Blackboard.

All the code for this homework should be in Scheme. Please submit one `.rkt` file on Blackboard.
In order to receive any credit for the programming portions, you are required to thoroughly comment and test your code.

1. **Mystery (2 points)**
   Consider the following function below:

   ```
   (define (mystery L)
     (if (= (length L) 0)
         '()
         (let ([k (quotient (length L) 2)])
           (list (list-ref L k) (mystery (take L k)) (mystery (drop L (+ k 1)))))))
   ```

   What does `(mystery '(1 2 3 4 5 6 7))` compute? As what kind of data structure can you interpret this? (Submit the answer for this question electronically on Blackboard; possibly as comments in the `.rkt` file.)

2. **Higher-Order Functions (4 points)**
   (a) (1 point) Implement a function `applyTwice` that applies a function `f` twice to an input argument `x`.
   (b) (2 points) Now implement a function `apply` that applies a function `f` `n`-times to an input argument `x`.
   (c) (1 point) Test `apply` and `applyTwice` by using a lambda expression to define an input function of your choice.

3. **Length (3 points)**
   Use `foldr` to define a function `myLength` that computes the length of an input list. Use a lambda expression to define the input function for `foldr`.

4. **Contains (3 points)**
   (a) (3 points) Use `foldl` or `foldr` to write a `contains` function that returns true if `x` is contained in the list `L`, and false otherwise.
   (b) **(2 extra credit points)** Use the `contains` function to implement a function `remove-duplicates` that removes all duplicates from an input list.

5. **Map (3 points)**
   Use `foldr` to define a `myMap` function that has the same functionality as `map`. Use a lambda expression to define the input function for `foldr`.

Flip over to back page ⟹

6. **Filter (5 points)**
   A unary predicate is a function that takes one argument and returns true or false. The built-in `filter` function takes a unary predicate `p` and a list `L` as arguments, and returns a list that contains all elements in `L` for which `p` is true.

   (a) (1 point) Write a predicate `isPositive` that returns true if the input is positive.

   (b) (1 point) Test the built-in filter function to return all the negative numbers from an input list of numbers. Use a lambda expression to define the predicate that returns true if the input is negative.

   (c) (3 points) Now use `foldr` to define a `myFilter` function that has the same functionality as `filter`. Use a lambda expression to define the input function for `foldr`.