CS302: Paradigms of Programming

Lab 4: Stream Processing

April 13th, 2021

Q1. In the class we had defined `(delay exp)` to be an abbreviation for the following:

```
(lambda () exp)
```

When we refer to a delayed argument many times in a procedure (say using the substitution model – which still works in our time-less and state-less stream-ful world :-)), an important optimization is to memoize the result of forcing a promise the first time it is referred to, and reuse it afterwards. One of the ways to impart memoization is to redefine `(delay exp)` to be an abbreviation for the following:

```
(memo-proc (lambda () exp))
```

where `memo-proc` is a procedure defined as:

```
(define (memo-proc proc)
  (let ((already-run? false) (result false))
    (lambda ()
      (if (not already-run?)
          (begin (set! result (proc))
                 (set! already-run? true)
                 result)
          result))))
```

First understand how `memo-proc` works, then use the concept of macros (`syntax-rules`) learnt today to define `delay` using `memo-proc`. Finally, write an example code that would benefit from memoization to improve efficiency in this manner (better if you can actually notice some difference while executing that example with and without memoization).

Q2. Define a procedure `partial-sums` that takes as argument a stream S and returns the stream whose elements are $S_0, S_0 + S_1, S_0 + S_1 + S_2, \ldots$. For example, recall the definition of the infinite streams of integers from class:

```
(define (integers-starting-from n)
  (cons-stream n (integers-starting-from (+ n 1))))
(define integers (integers-starting-from 1))
```

`(partial-sums integers)` should return the stream $1, 3, 6, 10, 15, \ldots$.

Q3. Recall the *Sieve of Eratosthenes* approach from the class. Explain the solution that we had studied to your partner(s) and to the TA. Now try to implement the same in a non-functional language that you know and notice the difference in expressiveness from our stream-based functional version. Appreciate the following quote:

"A good test of the expressive power of a programming language is the ease of implementing the Sieve of Eratosthenes algorithm in the same." – *Anonymous*