Quiz 10a

1. (1+2 points) We can also use a cyclic list to represent the infinite list of ones:

(define ones (list 1))

(set-cdr! ones ones)

(list-ref ones x) will give you back 1 for any valid x, so this really does represent an infinite list of ones. In this question, we explore some reasons why streams are better than cyclic lists.

* 1. Although we can represent the infinite list of ones, we cannot represent the infinite list of integers as a cyclic list. Explain why.
  2. Give an example of code using list functions that would cause an infinite loop when used with the cyclic representation, but the corresponding code using stream functions would work fine for an infinite stream of ones. For example, if you think that taking the car of the cyclic list would cause an infinite loop, but taking the stream-car of the stream of ones is fine, you should say (car ones). You may define helper procedures, but there are simple answers that do not require this. If your answer is wrong, you may get partial credit for an explanation of your answer.

1. (3 points) Here is an example of some very poorly written code – it depends on the implementation of streams (because it uses set!). As a result, for this question you should check and double-check your answers mechanically.

(define glob 1)

(define mystery

(cons-stream 1

(stream-map (lambda (x) (set! glob (+ glob 2))

(+ x glob))

mystery)))

After typing in these definitions, we execute the following code:

(stream-car (stream-cdr (stream-cdr mystery)))

(stream-car mystery)

* 1. What would the results be, assuming that the promises are memoized?

(stream-car (stream-cdr (stream-cdr mystery))) \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(stream-car mystery) \_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. What would the results be, assuming that the promises are not memoized? (Yes, there is a difference in the results of at least one of the expressions.)

(stream-car (stream-cdr (stream-cdr mystery))) \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(stream-car mystery) \_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (3+1 points) Let’s write a procedure to find the smallest factor of a number, apart from 1. We will use streams for this procedure. Here’s the algorithm to find the smallest factor of n:

Return the first element in the stream of the factors of n, which is found by filtering the elements of the stream of numbers between 2 and n.

1. Implement the procedure, using the algorithm above. Don’t use infinite streams.
2. Explain why we use streams instead of lists in this procedure. (It is not an issue with correctness.) Hint: Consider the input 1000000000.