Name: _____ Login: cs61a-___

QUIZ #3

1. (0.5 points) Write a procedure `stream-accumulate` that, given a binary operator and a stream as arguments, generates a stream of consecutive accumulations. For example, given an operator ♦ and a stream S, the elements of the resulting stream should be S_1, S_1 ♦ S_2, (S_1 ♦ S_2) ♦ S_3, …, where S_n is the n^{th} element of S.

   Use `stream-accumulate` to define the procedure `partial-sums`. Recall that `partial-sums` takes a stream S as an argument and generates a new stream in which the n^{th} element is $\sum_{i=1}^{n} S_i$.

   Use `stream-accumulate` to generate a stream of factorials. You may find it convenient to use `naturals`, an ordered stream of the positive integers.

Name: _____ Login: cs61a- ___

2. (0.5 points) Given the following code:

```
(define s1 (make-serializer))
(define s2 (make-serializer))

(define foo (list 'a 'b 'c))
```

Draw all possible box-and-pointer diagrams resulting from:

```
(parallel-execute (s1 (lambda () (set-car! foo (cdr foo)) ))
                  (s1 (lambda () (set-cdr! foo (car foo)) )) )
```

Draw all *additional* box-and-pointer diagrams if *instead of the above* we had:

```
(parallel-execute (s1 (lambda () (set-car! foo (cdr foo)) ))
                  (s2 (lambda () (set-cdr! foo (car foo)) )) )
```