CSSE 304, Winter 2024/25, Exam 2

Name:_____

By signing below, I certify that (i) all the work for this exam is my own work, (ii) I did not use any materials for the written portion and (iii) I did not use search engines or GenAI tools to look-up or produce answers for this exam. If you do not sign below, you will be assigned a zero (0) for the entire exam.

Your signature: _____

1) [10 pts] Implement a procedure called "compress-snlist" using snlist-recur. Compress-snlist takes an snlist as input and returns a new snlist where consecutive duplicate elements in any nested list are removed. The function should preserve the structure of the original snlist.

   Examples:

```
(compress-snlist '(1 1 2 2 3 3)) -> '(1 2 3)
(compress-snlist '((1 1 2) (2 2 3))) -> '((1 2) (2 3))
(compress-snlist '((1 (1 1 2)) (2 (2 2) 3))) -> '((1 (1 2)) (2 (2) 3))
(compress-snlist '()) -> '()
```

   In case you forgot the code for snlist-recur, here it is:

```
(define snlist-recur
  (lambda (flist fatom init)
    (letrec ([helper
              (lambda (l)
                (cond [(null? l) init]
                      [(pair? (car l))
                         (flist (helper (car l))
                                (helper (cdr l)))]
                      [else (fatom (car l)
                                   (helper (cdr l)))]))])
      helper)))
```

2) [5 pts] For our interpreter, we use nested environments. A faster approach would be to simply use a global hash-table. In it, much like for global defines, we have entries for each variable name. Each variable name is then associated with a list. If we have a new variable binding, we add the value to the beginning of this list. Please evaluate the benefits and drawbacks of this approach.

3) [5 pts] Evaluate the following lambda-app expression, drawing all closures and environments that are generated. Please enumerate the closures and environments to indicate the order in which they are generated. Please ensure that you make clear which environments are extended.

```
(((lambda (x)
    (lambda (x) (list x x))) 3) 4)
```

4) [10 pts] Evaluate the following lambda-app expression, drawing all closures and environments that are generated. Please enumerate the closures and environments to indicate the order in which they are generated. Please ensure that you make clear which environments are extended.

```
(((lambda (x)
    ((lambda (y) y)
     (lambda (z) z)))
  (lambda (a) a))
 42)
```