CM #1245

PLC A:19 - Written Portion

1. What is the output of the following code?

```
> (define mystery
    (let ([x 1] [y 2] [z 3])
      (let ([x 4] [b (+ 4 x)] [c (+ y y)])
        (let* ([x (* x x)] [d (* x x)])
          (apply + (list b c d x y z))))))
> mystery
```

Output: 286

-This questions tests a student's knowledge in the difference in environment access for let and let*.

2. Convert the following code to CPS. Assume you have the non-datatype implementation of apply-k and make-k.

```
>(define mystery2
    (lambda (ls max)
      (cond [(null? ls) max]
            [(null? max) (mystery2 (cdr ls) (car ls))]
            [(> (car ls) max) (mystery2 (cdr ls) (car ls))]
            [else (mystery2 (cdr ls) max)])))
>(mystery2 '(1 2 3) (list))
  3
```

Answer:

```
>(define mystery2
    (lambda (ls k)
      (cond [(null? (cdr ls)) (apply-k k (car ls))]
            [else (mystery2 (cdr ls) (make-k
                                      (lambda (x)
                                        (max (car ls) x))
                                      ))])))
> (mystery2 '(1 2 3) (make-k (lambda (v) v)))
  3
```

-Tests a student's knowledge of converting to CPS. Other answers may also work, this is just one of the possible solutions.