## General Instructions

Mutation, such as `set!` or `set-mcar!`, is generally poor style, so give at most a 3 to solutions using mutation.

## Problem 1

Here is a sample solution:

```
(define (racketlist->mupllist xs)
  (if (null? xs)
      (aunit)
      (apair (car xs) (racketlist->mupllist (cdr xs)))))

(define (mupllist->racketlist xs)
  (if (aunit? xs)
      null
      (cons (apair-e1 xs) (mupllist->racketlist (apair-e2 xs)))))
```

It is unlikely that there are many ways to make solutions much longer or more complicated while having style as good as the sample solution, so mostly follow general guidelines.

Solutions using higher-order functions, such as `foldl` or `foldr` from Racket's standard library, can get a 5 if they are clear and concise.

Some students may have misinterpreted the problem as requiring recurring into nested lists (even though the problem said to presume the list contents were MUPL values and therefore shouldn't be changed). This can lead to more complicated and unnecessary code, but from a style perspective, there is no need to give significant penalties if this is the only complication, so otherwise good solutions would probably deserve a 4.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

## Problem 2

**For problem 2, we will break the assessment down into more manageable pieces with separate grades for each piece.**

## Problem 2: Overall Structure

Here evaluate if their solution has the right overall structure of having one case for each kind of MUPL expression. Some helper functions outside the big cond expression are okay although the sample solution does not have any.

```
(define (eval-under-env e env)
  (cond [(var? e) ...]
        [(int? e) ...]
        [(add? e) ...]
        [(ifgreater? e) ...]
        [(fun? e) ...]
        [(call? e) ...]
        [(mlet? e) ...]
        [(apair? e) ...]
        [(fst? e) ...]
        [(snd? e) ...]
        [(aunit? e) ...]
        [(isaunit? e) ...]
        [(closure? e) ...]
        [#t ...]))
```

Give a 3 or a 4 to solutions that do not follow this structure depending on how clear the overall structure is.

The order of cases does not matter.