CSSE 304 Exam #2 E&C Fall 202210 Name KEY Section: 02(11:00) 03(3:00)

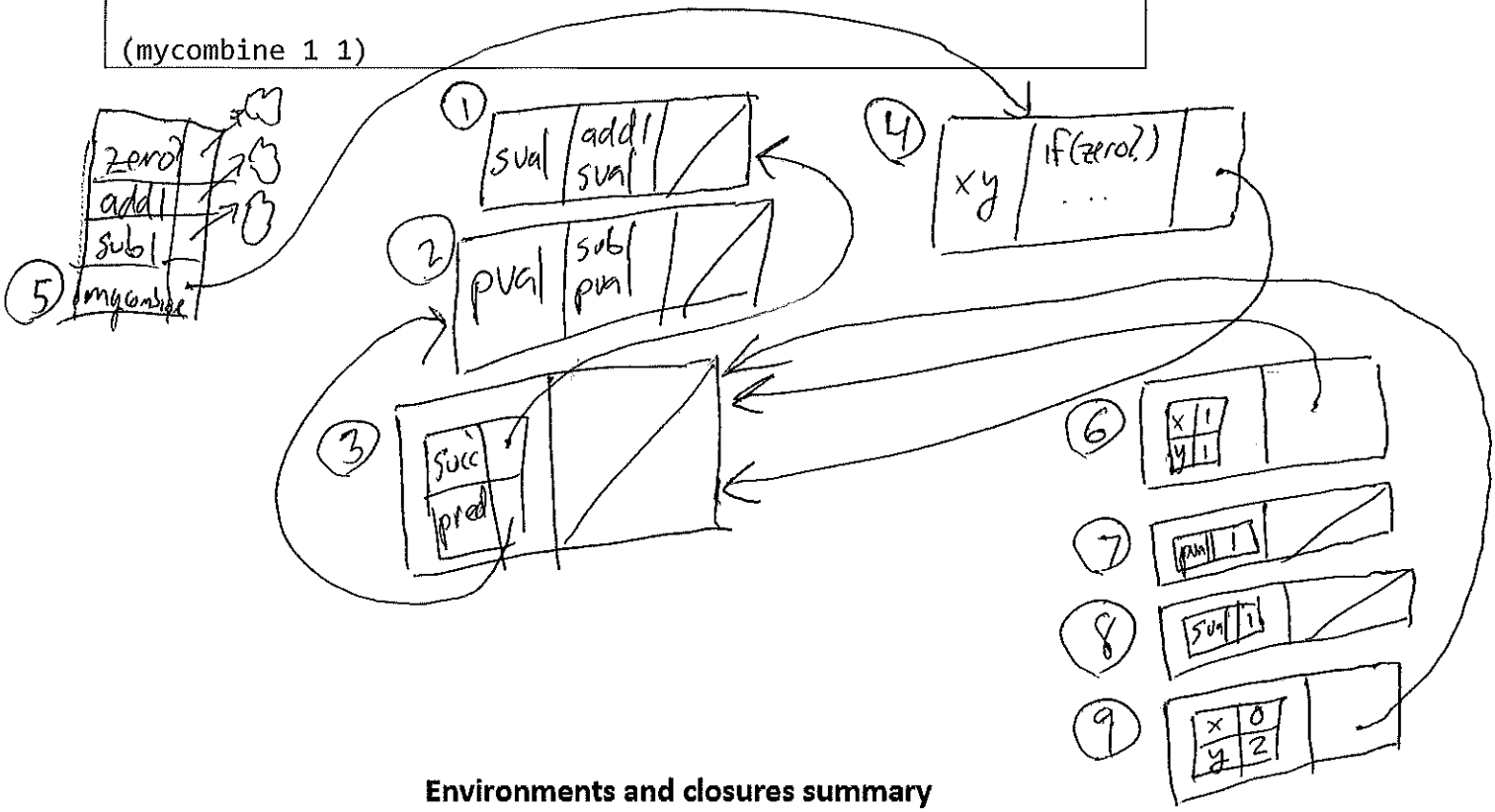
(20 points) Draw the environments and closures created during the execution of the following code segment. Include a sequence number for each new environment, new closure, or global environment change.

If you need to start over, use the back, and put a big X through what you wrote here.

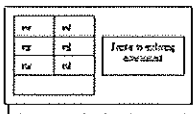
Value returned by the expression: _____

```
(define mycombine
    (let ([succ (lambda (sval) (add1 sval))]
          [pred (lambda (pval) (sub1 pval))])
        (lambda (x y)
            (if (zero? x) y (mycombine (pred x) (succ y))))))

(mycombine 1 1)
```



Environments and closures summary

Environment: 

Closure:

list of formal argument names	code (body of the procedure)	local environment that existed when the procedure was created

A procedure (a.k.a. **closure**) is created when a `lambda` expression is evaluated. The body of the closure is not evaluated at this time.

Application of a closure (user-defined procedure):

1. The expressions for the procedure and its arguments are evaluated.
2. A new local environment is created.
   a. Each variable from the procedure's formal parameter list is bound to the corresponding value in the actual argument list.
   b. The new environment's "pointer to an enclosing environment" is set to be a copy of the local environment pointer **that is the third part of the closure.**
3. The body of the procedure is evaluated, using this new local environment. If a variable is not found in this local environment or something it points to, look in the global environment. If not in global environment either, it is an error.

Evaluate a `let` expression:

1. Evaluate (in the current environment) the expressions to get the values to be assigned to the `let` variables.
2. Create a new local environment with bindings for the let variables. The "enclosing environment" pointer points to the current environment.
3. Evaluate the body of the `let` in this new environment, as in 3 above.

Evaluate a `letrec` expression:

1. Create a new local environment, similar to a `let` environment, except that:
   a. The "saved environment" pointers of any closures that are bound to the `letrec` variables point to the new letrec environment, not the enclosing environment.
2. Evaluate the body of the `letrec` in this new environment, as in 3 above.