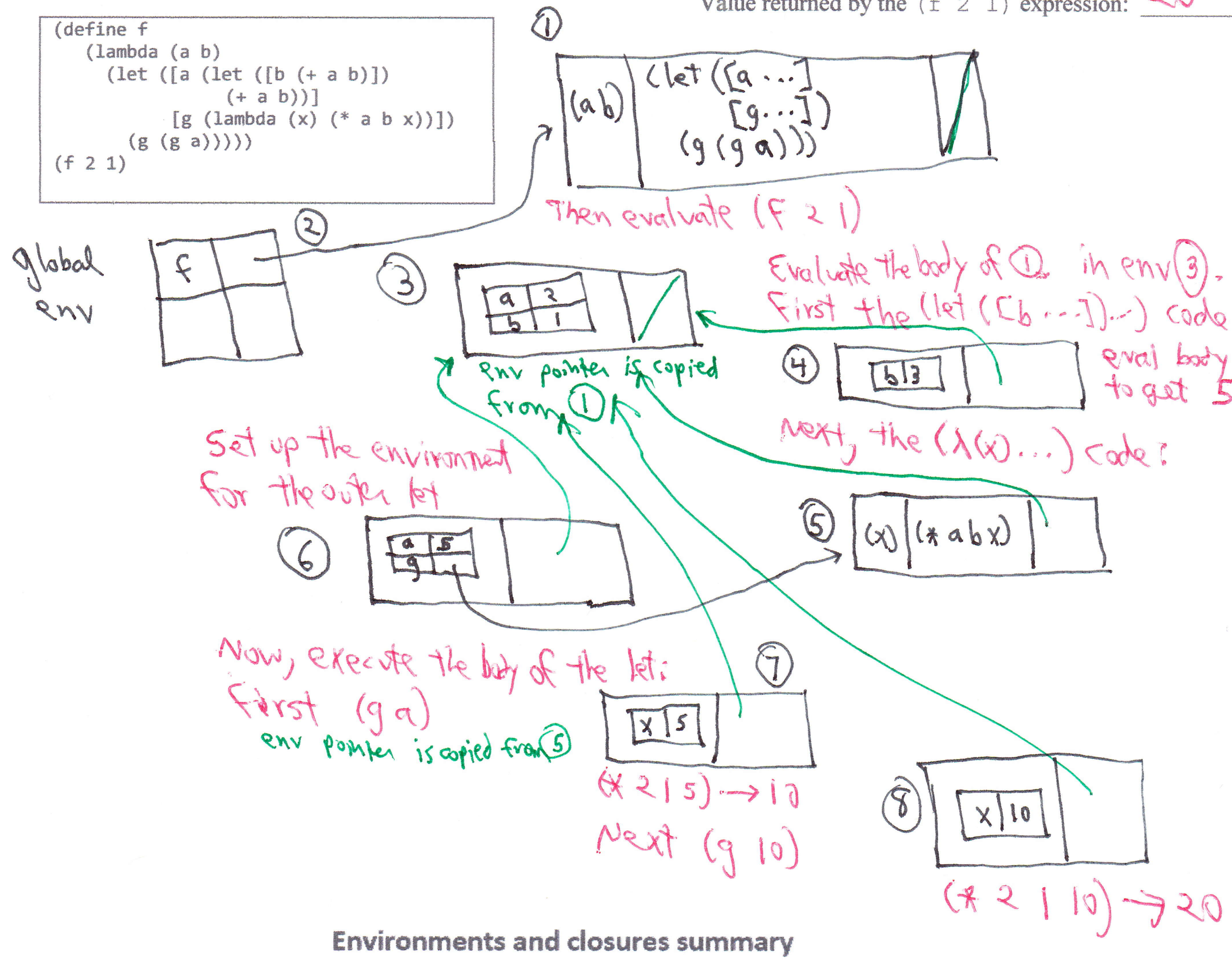
CSSE 304 Exam #2 E&C Jan 17, 2019 (day 23) Name Solution **Section:** 01(9:00) 02(9:55) 03(10:50)

This space is here in case you need to start over in order to avoid having a messy diagram. If you want me to grade this side, place a bog X through your work on the other side.

Value returned by the `(f 2 1)` expression: 20

```
(define f
   (lambda (a b)
     (let ([a (let ([b (+ a b)])
                 (+ a b))]
           [g (lambda (x) (* a b x))])
        (g (g a)))))
(f 2 1)
```



Then evaluate (f 2 1)

Evaluate the body of ① in env ③. First the (let ([b ...]) ...) code

eval body of let to get 5.

Next, the (λ(x) ...) code:

env pointer is copied from ①

Set up the environment for the outer let

Now, execute the body of the let: First (g a)

env pointer is copied from ⑤

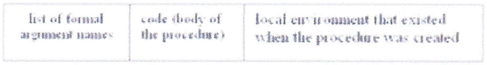
$(* 2\ 1\ 5) \rightarrow 10$

Next (g 10)

$(* 2\ 1\ 10) \rightarrow 20$

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.