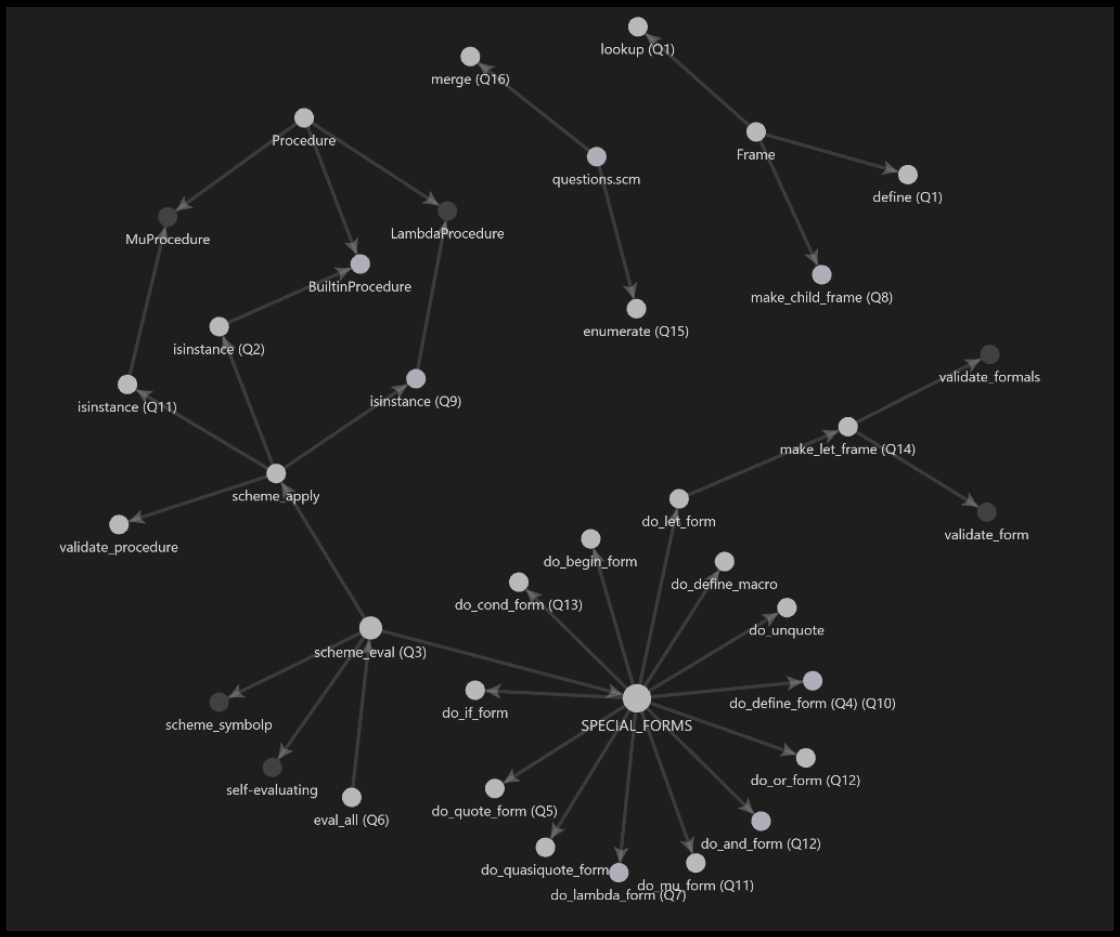

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Python

```
          '()
          (cons (* (car lst) 2) ((lambda (f) (f (cdr lst))) (lambda (x) x))))))
(2 2 3 4 5)
scm> ((lambda (lst)
       (if (null? lst)
           '()
           (cons (* (car lst) 2) ((lambda (f) (f (cdr lst))) (lambda (x) x)))))
     (list 1 2 3 4 5))
(2 2 3 4 5)
scm> (apply + (map (lambda (x) (* x x)) '(1 2 3 4 5)))
55
scm>
```

scheme Copy

```scheme
(define (fib n)
  (if (< n 2)
      n
      (+ (fib (- n 1)) (fib (- n 2)))))

(map fib (iota 10))
```

```
def do_define_form(expressions, env):
    """Evaluate a define form.
    >>> env = create_global_frame()
    >>> do_define_form(read_line("(x 2)"), env) # evaluating (define x 2)
    'x'
    >>> scheme_eval("x", env)
    2
    >>> do_define_form(read_line("(x (+ 2 8))"), env) # evaluating (define x (+ 2 8))
    'x'
    >>> scheme_eval("x", env)
    10
    >>> # problem 10
    >>> env = create_global_frame()
    >>> do_define_form(read_line("((f x) (+ x 2))"), env) # evaluating (define (f x) (+ x 8))
    'f'
    >>> scheme_eval(read_line("(f 3)"), env)
    5
    """
    validate_form(expressions, 2)  # Checks that expressions is a list of length at least 2
    signature = expressions.first
    if scheme_symbolp(signature):
        # assigning a name to a value e.g. (define x (+ 1 2))
        validate_form(expressions, 2, 2)  # Checks that expressions is a list of length exactly 2
        # BEGIN PROBLEM 4
        f=expressions.rest.first
        func=scheme_eval(f, env)
        env.define(signature,func)
        return signature
        # END PROBLEM 4
    elif isinstance(signature, Pair) and scheme_symbolp(signature.first):
        # defining a named procedure e.g. (define (f x y) (+ x y))
        # BEGIN PROBLEM 10
        symbol=signature.first
        formals=signature.rest
        body=expressions.rest
        validate_formals(formals)
        product=LambdaProcedure(formals,body,env)
        env.define(symbol,product)
        return symbol
        # END PROBLEM 10
    else:
        bad_signature = signature.first if isinstance(signature, Pair) else signature
        raise SchemeError('non-symbol: {0}'.format(bad_signature))
```