Exam 3

● Graded

Student

Adam Fenjiro

Total Points

29 / 70 pts

Question 1

Q1 **4** / 4 pts

1.1 a **1** / 1 pt

✓ **– 0 pts** Correct

– 1 pt Incorrect (Answer: 30)

1.2 b **1** / 1 pt

✓ **– 0 pts** Correct

– 0.5 pts Extra parentheses

– 1 pt Incorrect (should be `'Congrats`)

1.3 c **1** / 1 pt

✓ **– 0 pts** Correct

– 0.5 pts Missing parentheses

– 1 pt Incorrect (Answer: `'(of 2025)`)

1.4 d **1** / 1 pt

✓ **– 0 pts** Correct

– 0.5 pts One extra pair of parentheses OR one pair of parentheses missing

– 1 pt Incorrect (Answer: `'((Congrats) Class of 2025)`)

Question 2

Q2 **3** / 5 pts

2.1 a **1** / 1 pt

✓ **− 0 pts** Correct

− 0.5 pts Missing one first or rest

− 1 pt Incorrect (Answer: `(first (rest (rest (rest '(Scheme is a functional language)))))`)

2.2 b **1** / 1 pt

✓ **− 0 pts** Correct

− 0.5 pts Missing one first or rest

− 1 pt Incorrect (Answer: `(first (first (rest (rest (rest '((Scheme) (is) (a) (functional) (language)))))))`)

2.3 c **0.5** / 1 pt

− 0 pts Correct

✓ **− 0.5 pts** Missing one first or rest

− 1 pt Incorrect (Answer: `(first (first (rest '((Scheme is a) (functional language)))))`)

2.4 d **0.5** / 1 pt

− 0 pts Correct

✓ **− 0.5 pts** Missing one first or rest

− 1 pt Incorrect (Answer: `(first (rest (first '(((((Scheme) is) a) functional) language))))`)

2.5 e **0** / 1 pt

− 0 pts Correct

− 0.5 pts Missing one first or rest

✓ **− 1 pt** Incorrect (Answer:
`(first (first (rest (first  (rest (first (rest '(Scheme (is (a (functional (language))))))))))))`)

Question 3

Q3 **4** / 5 pts

3.1 a **1** / 1 pt

✓ **– 0 pts** Correct

– 0.5 pts Missing one cons

– 1 pt Incorrect (Answer: `(cons 'a (cons 'b (cons 'c '())))`)

3.2 b **1** / 1 pt

✓ **– 0 pts** Correct

– 0.5 pts Missing one cons OR one extra cons

– 1 pt Incorrect (Answer: `(cons (cons 'a (cons 'b '())) (cons (cons 'c '()) '()))`)

3.3 c **0.5** / 1 pt

– 0 pts Correct

✓ **– 0.5 pts** Missing one cons OR one extra cons

– 1 pt Incorrect (Answer: `(cons (cons (cons 'a '()) (cons 'b '())) (cons 'c '()))`)

3.4 d **1** / 1 pt

✓ **– 0 pts** Correct

– 0.5 pts Missing one cons OR one extra cons

– 1 pt Incorrect (Answer: `(cons 'a (cons (cons 'b (cons (cons 'c '()) '())) '()))`)

3.5 e **0.5** / 1 pt

– 0 pts Correct

✓ **– 0.5 pts** Missing one cons OR one extra cons

– 1 pt Incorrect (Answer: `(cons (cons 'a (cons 'b '())) (cons (cons 'c (cons '()'())) '()))`)

The actual error is the position of 'c, but since this was a rare error I didn't make a specific rubric for this.

Question 4

Q4 **3** / 3 pts

✓ **– 0 pts** Correct

– 0.5 pts Did not use define

– 1 pt Prototype incorrect (cal x y)

– 0.5 pts Order of operations incorrect

– 1.5 pts Function body incorrect

– 3 pts Not attempted or incorrect

Question 5

Q5 **3** / 3 pts

✓ **– 0 pts** Correct

– 1 pt lambda not used correctly

– 0.5 pts Prototype incorrect (x y)

– 0.5 pts Order of operations incorrect

– 1.5 pts Function body incorrect

– 3 pts Incorrect OR did not use lambda at all

Question 6

Q6 **3** / 6 pts

6.1 **a** **1** / 3 pts

– 0 pts Correct

– 1 pt Did not use map and cal

– 0.5 pts Minor error in body

✓ **– 2 pts** Function body incorrect

– 3 pts Incorrect

6.2 **b** **2** / 3 pts

– 0 pts Correct

– 0.5 pts Null check incorrect

– 0.5 pts Missing cons (Returned value needs to be a list)

– 0.5 pts Other minor function body error

– 1 pt Used cal incorrectly

✓ **– 1 pt** Recursion incorrect

– 3 pts Incorrect

Question 7

Q7 **4.5** / 5 pts

– 0 pts Correct

✓ **– 0.5 pts** Minor mistake in any part

– 1 pt Null check incorrect (terminating condition)

– 2 pts Non-recursive part incorrect

– 2 pts Recursive part incorrect

– 5 pts Incorrect

Question 8

Q8

0.5 / 4 pts

- **− 0 pts** Correct
- **− 0.5 pts** Minor error in any item
- **− 1 pt** Recursive calls missing (used (rest M) directly instead)
- **− 0.5 pts** Null check incorrect (terminating condition)
- ✓ **− 1.5 pts** Call when (first M) is a list incorrect
- ✓ **− 1.5 pts** Call when (pred (first M)) is true incorrect
- ✓ **− 0.5 pts** Call when (pred (first M)) is false incorrect
- **− 4 pts** Incorrect

Question 9

Q9

1 / 5 pts

- **− 0 pts** Correct
- **− 0.5 pts** Minor error in any step
- **− 1 pt** Null check (termination condition) incorrect
- ✓ **− 2 pts** Call when (first M) is a pair is incorrect
- ✓ **− 2 pts** Call when (first M) is an atom is incorrect
- **− 5 pts** Incorrect

Question 10

Q10

0 / 3 pts

- **− 0 pts** Correct
- **− 0.5 pts** Minor error in any step
- ✓ **− 1 pt** Initial value incorrect
- **− 1 pt** Function passed to map incorrect
- ✓ **− 2 pts** Map expression (or equivalent) incorrect
- **− 3 pts** Incorrect

Question 11

Q11 1 / 4 pts

- − **0 pts** Correct
- − **0.5 pts** Minor error
- − **1 pt** Initial values incorrect
- − **1 pt** Function passed to map incorrect
- ✓ − **2 pts** Map expression incorrect
- ✓ − **1 pt** Prior terms in the sequence incorrect
- − **4 pts** Incorrect

Question 12

Q12 1 / 4 pts

- − **0 pts** Correct
- − **0.5 pts** Minor error in any step
- − **1 pt** Abstraction interface incorrect
- ✓ − **2 pts** Abstraction body incorrect
- ✓ − **0.5 pts** add-pairs redefinition incorrect
- ✓ − **0.5 pts** sub-pairs redefinition incorrect
- − **4 pts** Incorrect

Question 13

Q13 0 / 7 pts

- − **0 pts** Correct
- − **0.5 pts** Minor error in any step
- − **1 pt** Uses local incorrectly
- − **2 pts** firstcolumn implementation incorrect
- − **2 pts** restcolumns implementation incorrect
- − **2 pts** Main transpose body implementation incorrect
- ✓ − **7 pts** Incorrect

Question 14

Q14 **0** / 6 pts

- **– 0 pts** Correct
- **– 2 pts** a) Incorrect
- **– 2 pts** b) Incorrect
- **– 2 pts** c) Incorrect
- **– 2 pts** Vtable not shown separately from object layout (vtable is per class, not per object)
- **– 2 pts** An object is shown to have multiple vtables
- **– 1 pt** Explanation insufficient or partly incorrect for c
- ✓ **– 6 pts** Not attempted

Question 15

Q15 **1** / 6 pts

15.1 a **0** / 1 pt

- **– 0 pts** Correct
- ✓ **– 1 pt** Incorrect (Marks live objects reachable from the root set)

15.2 b **0** / 1 pt

- **– 0 pts** Correct
- ✓ **– 1 pt** Incorrect (Scan the heap and reclaim all unmarked (dead) objects)

15.3 c **0** / 2 pts

- **– 0 pts** Correct
- **– 1 pt** Yes/no answer incorrect (No.)
- **– 1 pt** Why answer incorrect (All ref. counts remain at least one due to internal links.)
- ✓ **– 2 pts** Incorrect

15.4 d **1** / 2 pts

- **– 0 pts** Correct
- **– 1 pt** Yes/no answer incorrect (Yes.)
- ✓ **– 1 pt** Why answer incorrect (The whole list can become unreachable and thus dead.)
- **– 2 pts** Incorrect

CS4121 Exam #3
April 23, Spring 2025

Name: Adam FENJIRO User ID: afenjiro
(User ID is your Michigan Tech email ID. For example, put in *joe* if your email address is jeo@mtu.edu.)

1. (4 pts) Evaluate the following Scheme expressions.

   (a) `(+ 1 (* 2 3 4) 5)`

   = (+ 1 24 5) = 30

   (b) `(first '(Congrats Class of 2025))`

   Congrats

   (c) `(rest (rest '(Congrats Class of 2025)))`

   = (rest '(Class of 2025)) = (of 2025)

   (d) `(cons '(Congrats) '(Class of 2025))`

   = ((Congrats) Class of 2025)

2. (5 pts.) For each of the following Scheme lists, write an expression using only **first** and **rest** that will return the element `functional` when applied to the list. You can use L to represent the list in your answer.

   (a) `'(Scheme is a functional language)`

   (first (rest (rest (rest '(L)))))

   (b) `'((Scheme) (is) (a) (functional) (language))`

   (first (first (rest (rest (rest '(L))))))

   (c) `'((Scheme is a) (functional language))`

   (first (rest '(L)))

   (d) `'(((((Scheme) is) a) functional) language)`

   (rest (first '(L)))

   (e) `'(Scheme (is (a (functional (language)))))`

   (first (rest (rest '(L))))

3. (5 pts.) Using only the symbols 'a, 'b, and 'c, the Scheme function cons and the null list, write a Scheme expression that constructs the following lists.

(a) '(a b c)

```
(cons 'a (cons 'b (cons 'c ())))
```

(b) '((a b) (c))

```
       (a b)                           ((c))
(cons (cons 'a (cons 'b ())) (cons (cons 'c ())))
```

(c) '(((a) b) c)

```
                 ( (a)   b)                      c
(cons (cons (cons (cons 'a ()) ) (cons 'b ()) (cons 'c ()))
```

(d) '(a (b (c)))

```
(cons 'a (cons (cons 'b (cons (cons 'c ())))))
```

(e) '((a b) (c ()))

```
(cons (cons 'a (cons 'b ())) (cons 'c (cons (cons ()))))
```

4. (3 pts.) Write a function `cal` that takes two numbers `x` and `y` and returns `2*x - y`.

```
(cal 3 4) --> 2
```

```
(define cal (x y)
(- (*2 x) y))
```

5. (3 pts.) Write a **no name** function that takes two numbers `x` and `y` and returns `2*x - y`.

```
(lambda (x y) (- (*2 x) y))
```

6. (6 pts.) Write a function **calList** that takes a flat list of numbers and returns a flat list of numbers that doubles each element of the input list and then subtracts it by 1, respectively.

```
(calList '(1 -2 3 9)) --> '(1 -5 5 17)
```

(a) (3 pts.) Define **calList** using **map** and **cal** defined in Problem 4.

```
(define calList (L)
mep ( cal (L 1)))
```

(b) (3 pts.) Define **calList** using **cal** defined in Problem 4 but you are not allowed to use **map**.

```
(define calList (L) (cond
 [(null? L) L]
 [else (cons (- (* first(L) 2) 1) (rest(L)))]
))
```

7. (5 pts.) Write a function **interleave** that takes two flat lists **L1** and **L2** of the same length and generates a list with the elements from **L1** and **L2** interleaved.

```
(interleave '(a b c d) '(1 2 3 4)) --> '(a 1 b 2 c 3 d 4)
```

```
(define interleave (L1 L2) (cond
   [(null? L1) L2]   ; if L1 null, we return L2
   [(null? L2) L1]   ; if L2 null, we return L1
else
[cons (first (L1)) (cons ((first (L2)) (rest (L1) (rest (L2)))))]
))
```

8. (4 pts.) Write a function `find-pred` that takes a predicate `pred` and a list `M`, and returns a flat list of all the atoms contained in `M` that satisfy `pred`. You might find the `append` function useful and that it exists. See below for example usage.

```
; append example
(append '(a b c) '(d e f)) --> '(a b c d e f)

; function example
(find-pred number? '(a ((2) 3) b (2) (c) 1 ())) --> '(2 3 2 1))
```

```
(define find-pred (M pred) (cond
  [(null? M) M]
  [(pred? M) append
                      (cons (first'(M)) (rest'(M)))]
))
```

9. (5 pts.) Write a function `clone` that takes a list `M`, and returns a list with each atom duplicated.

```
(clone '(a (()) (c d))) --> '(a a (() ()) (c c d d))
```

```
(define clone (M) (cond
  [(null? M) M]
 [else  (cons (first'(M)) (cons (first'(M)) (rest'(M))))]
))
```

10. (3 pts.) Using Lazy Scheme, write an infinite list of negative integers, **negints**.

```
(!!(take 5 negints)) --> (-1 -2 -3 -4 -5)
```

```
(define negints (L) (cond
    [(null? L) L]
   [else (cons (- first'(L) (* first'(L) 2)) (rest'(L) )))]
))
```

11. (4 pts.) Using Lazy Scheme, write an infinite list, `f-list`, that contains all solutions to the following recurrence relation:

$$f(0) = 0$$
$$f(1) = 1$$
$$f(n) = 2 * f(n-1) + f(n-2), \quad n > 1$$

```
(!!(take 7 f-list)) --> (0 1 2 5 12 29 70)
```

```
(define f-list( n) (cond
 [(eq? n 0) 0]
 [(eq? n 1) 1]
 [(>? n 1) (+ (* 2 (-n 1)) (- n 2))]
))
```

12. (4 pts.) Create a functional abstraction for the following two functions and redefine each function in terms of the abstraction.

```
(define (add-pairs P)
  (cond
     [(null? P) '()]
     [(null? (rest P)) P]
     [else (cons (+ (first P) (first (rest P))) (add-pairs (cddr P)))]
   )
)

(define (sub-pairs P)
  (cond
     [(null? P) '()]
     [(null? (rest P)) P]
     [else (cons (- (first P) (first (rest P))) (sub-pairs (cddr P)))]
  )
)
```

```
              P,f
(define abs ( )(cond
   [(null ? P) '()]
   [(null ? (rest P)) P]
  ~~[(eq? f 'add-pairs) (cons (+ (first P) (first (rest P))) (add-pairs add)~~
  [(eq? f 'add-pairs) add-pairs (P)]
  [(eq? f 'sub-pairs) sub-pairs (P)]
))
```

13. (7 pts.) Write a function (**transpose** M) that returns the transpose of matrix M, which is represented as a list of rows, each of which is a flat list of numbers.

```
(transpose '((1 2 3) (4 5 6))) --> '((1 4) (2 5) (3 6))
```

Your answer must contain two local functions (**firstcolumn** M) and (**restcolumns** M) which returns the first column and the rest columns of M, respectively.

```
(firstcolumn '((1 2 3) (4 5 6))) --> '(1 4)
```

```
(restcolumns '((1 2 3) (4 5 6))) --> '((2 3) (5 6))
```

You can then implement (**transpose** M) using these two local functions.

14. (6 pts.) You are given the Java program below.

(a) (2 pts.) If we run the program with argument "*P*", lines A and C would be executed. Show the object layout for variable o including the vtable when line C is executed.

(b) (2 pts.) If we run the program with argument "*C*", lines B and C would be executed. Show the object layout for variable o including the vtable when line C is executed.

(c) (2 pts.) Explain in a couple sentences how the compiler can generate code for line C so the right method will be called for both case (a) and case (b).

```
class Parent {
    private int age;
    protected String last;
    public Parent(int a, String l)
    {
      age = a;
      last = l;
    }
    public int getAge()
    {
      return age;
    }
    public void printInfo()
    {
      System.out.println(last+": "+getAge());
    }
}
class Child extends Parent {
    private String first;
    public Child(int a, String l, String f)
    {
      super(a, l);
      first = f;
    }
    public void printInfo()
    {
      System.out.println(last+" "+first+": "+getAge());
    }
}

public class SingleInheritance {
    public static void main(String args[])
    {
      Parent p = new Parent(36, "Smith");
      Child c = new Child(4, "Smith", "Joe");
      Parent  o;

      if (args[0].compareTo("P")==0)  // if the first argument is "P"
        o = p;                        // line A
      else
        o = c;                        // line B

      o.printInfo();                  // line C
    }
}
```

Space for Question 14

15. (Bonus: 6 pts.) Compare the reference counting garbage collector against the mark-sweep garbage collector. Specifically, answer the following questions.

(a) (1 pt) What does the mark phase of mark-sweep do?

(b) (1 pt) What does the sweep phase of mark-sweep do?

(c) (2 pts) Can reference counting be used to recycle a cyclic linked list? Why?

Yes, it can.

(d) (2 pts) Can mark-sweep be used to recycle a cyclic linked list? Why?

Yes, it can.