

CS51 Assignment 5: Midterm Review

Due: Tuesday, March 17th, 2009, 5:00PM

Total Points: 46

This assignment is a review of several of the concepts covered in class so far, and is meant to mimic some of the questions you might see on the midterm. Although you can test many of your functions by running them on your computer, be aware that Dr. Scheme won't be available on the midterm. Good luck studying!

1 Evaluation Rules

(10 points total)

Exercise 1.

7 points

`(define f _____)`

For each of the following expressions, fill in a definition of `f` such that the value of the following expressions is 42, or state that this is impossible and explain why.

(a) [1 point] `(f)`

(b) [2 points] `((f 1) 2 3)`

(c) [2 points] `(f 42 (f 42))`

(d) [2 points] `(+ (f (f 2)) (f 2))`

Exercise 2.

3 points

(a) [3 points]

Your good friend John Scheme is convinced that the builtin Scheme function, `or`, is too slow. He decides to define his own `or` function. You inform your friend that this is not a good idea, and that `my-or` will behave differently than `or`. What is the problem? As part of your explanation, you should include an example usage of `my-or` that is not equivalent to the same expression with `or`.

```
(define (my-or val1 val2) (if val1 #t val2))
```

2 Fun With Lists

*(5 points total)***Exercise 3.***5 points*

- (a) [2 points] Write `my-reverse-1` which reverses a list using the `append` function.
- (b) [2 points] Write `my-reverse-2` which reverses a list without using the `append` function. Depending on how you approach this, a helper function may be useful, but is not necessary.
- (c) [1 point] Which of the two reverse functions you just wrote is generally preferable, and why? Give your answer in three sentences or less.
- (d) [3 points] What does the following function `mystery` do? What is its running time? Assume `lst1` and `lst2` are lists.

```
(define (mystery lst1 lst2)
  (if (empty? lst1)
      0
      (+ (mystery (cdr lst1) lst2)
         (mystery-helper (car lst1) lst2))))

(define (mystery-helper e1 lst2)
  (cond [(empty? lst2) 0]
        [(= (+ e1 (car lst2)) 5) (+ 1 (mystery-helper e1 (cdr lst2)))]
        [else (mystery-helper e1 (cdr lst2))]))
```

- (e) [3 points] Inductively prove that the function `num-even` returns the number of even integers in a list.

```
(define (num-even lst)
  (cond [(empty? lst) 0]
        [(even? (car lst)) (+ 1 (num-even (cdr lst)))]
        [else (num-even (cdr lst))]))
```

3 Higher Order Functions

*(14 points total)***Exercise 4.***14 points*

- (a) [6 points] Write `foldr` and `foldl`.
- (b) [2 points] Write `map` in terms of `foldr` or `foldl`.
- (c) [2 points] Write `filter` in terms of `foldr` or `foldl`.
- (d) [2 points] Write `sum-deep`, which sums all the numbers in an arbitrarily nested list. Namely `(sum-deep (list 1 2 (list 3 (list 4))))` should evaluate to 10. You should

use one or more of the above functions.

(e) [2 points] Write `random-eval`, which takes two functions and an alpha, and returns a function which will with probability alpha return the evaluation of the first function on no arguments, otherwise it returns the evaluation of the second function on no arguments.

4 Graph Search

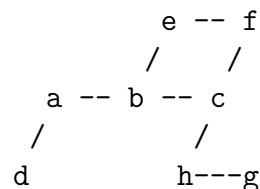
(6 points total)

Exercise 5.

6 points

For the following questions you will navigate a graph using the algorithms we have learned in class. All searches begin at node `a`, and assume ties are broken in alphabetical order.

Consider the following graph:



(a) [2 points] List in order the nodes which BFS will visit when searching for a node that is not in the graph.

(b) [2 points] List in order the nodes which DFS will visit when searching for a node that is not in the graph.

(c) [2 points] List in order the nodes which Iterative Deepening Search will visit when searching for the node `f`. (Note that it is *not* visiting all the nodes in the graph).

5 Modules and Contracts

(9 points total)

Exercise 6.

3 points

Given the following struct, what methods does PLT Scheme automatically provide?

```
(define-struct test (questions score passed?))
```

Exercise 7.

6 points

We've included an implementation of a priority queue in the `asst5.scm` file. A priority queue is an ADT that provides three operations – `insert`, `get-min`, and `remove-min`. `insert` adds an integer to the priority queue, `get-min` returns the minimum element in the priority queue, and `remove-min` removes the minimum element from the priority queue.

(a) [2 points] Comment on the performance of this module – specifically, what is the Big-O runtime of each of the three operations?

(b) [2 points]

John Scheme is back and argues that `insert` should insert the numbers in a list in order and `get-min` and `remove-min` should get and remove elements from the front of the list, respectively. He claims that this would be faster – is he right? You should include both asymptotic and practical considerations in at most five sentences.

(c) [2 points]

Finally, consider the following scenario: we are going to insert the integers 1 through 5 into the priority queue and then empty the queue by calling `get-min` followed by `remove-min` five times. Give two orderings of inserts that give the best and worst case performance of John’s priority queue. Is there an ordering for which the included implementation performs better than John’s?

6 Finishing Up

(2 points total)

Exercise 8.

2 points

(a) [1 point] Please write the following (true) sentence: “I have completed my midterm course and section evaluations. [Your Name Here]”

(b) [1 point]

Please tell us how much time you spent on this problem set. You may use this space to add any thoughts or questions you want to send to your TF as well, if you so desire.