"Scheming Sets"

Programming Lab 3

Concepts of Programming Languages
CSCI305, Spring 2015

Due: April 17, 2015 at 11:59 pm

Scheme

For this lab, you will use Scheme. You may use any flavor of scheme of your choice. I recommend MIT/GNU Scheme `http://www.gnu.org/software/mit-scheme/` or Racket `http://racket-lang.org/`.

Warmup

Begin by entering this function in Scheme. The lines that begin with a semicolon are comment lines that you will fill in.

```
(define (f lst)
  ; (a) ;
  (if (null? lst)
      ; (b) ;
      '()
      ; (c) ;
      (cons (+ 1 (car lst))  (f (cdr lst)))))
```

Lab Questions

Question 1: Run this function as `(f '(3 1 4 1 5 9))`. What output do you get?

Question 2: What does this function `f` do?

Question 3: Give a comment that explains the line following `(a)`.

Question 4: Give a comment that explains the line following `(b)`.

Question 5: Give a comment that explains the line following `(c)`.

Question 6: Trace the call given in Question 1, showing each recursive call to the function. Specifically show each call, expanding the term `lst` to specific lists or atoms at each depth of the recursion.

Member? function

Write a function `member?` that determines if an element `e` is part of list `lst`. This function will return `#t` if `e` is a member of the list `lst` and `#f` otherwise. You may use common Scheme functions (e.g., `car, cdr, cons, eq?, eqv?, equal?, null?,` and `list?`). If your Scheme implementation has a builtin member function, you may **not** use it in your answer. Comment your function.

```
(define (member?  e lst)


   ; complete this function definition


)
```

Set? function

Write a function `set?` that checks whether its argument `lst` is a well formed set, that is a set with no duplicates, and returns true `#t` or false `#f` accordingly. You may find it useful to make use of your `member?` function in your function.

For example:

```
(set?  (x y z)) => #t
(set?  (a 1 b 2 c 3)) => #t
(set?  ()) => #t ; empty set is a good set
(set?  (6 2 2)) => #f ; duplicate, bad set
(set?  (x y z x)) => #f ; duplicate, bad set
```

```
(define (set?  lst)

   ; complete this function definition

)
```

Lab Questions

Your answers **must** reflect the output of your code. No credit will be given to answers if you have not submitted the respective correct function implementation.

Question 7: What output do you get for the call `(member?  'one '(1 2 3 4))`

Question 8: Does your `member?` function use head or tail recursion?

Question 9: What output do you get for the call

```
(set?  '(it was the best of times, it was the worst of times))
```

Question 10: Research `tail recursion`. Describe, in a few short sentences, why it can be beneficial to write tail recursing functions.

Union function

Write a function `union` that takes the set union of list `lst1` and list `lst2` and returns a list representing the mathematical union of the two lists. Your function does not need to work on embedded lists. You may use the functions you defined previously (`set?` and `member?`), if useful, in addition to the common Scheme functions mentioned above. If your Scheme implementation has a union function, you may **not** use it in your answer. Comment your function.

```
(define (union lst1 lst2)

   ; complete this function definition

)
```

Intersect function

Write a function `intersect` that takes the set intersection of list `lst1` and list `lst2` and returns a list representing the mathematical intersection of the two lists. Your function does not need to work on embedded lists. If your Scheme implementation has a intersection function, you may **not** use it in your answer. Comment your function.

```
(define (intersect lst1 lst2)

   ; complete this function definition

)
```

Lab Questions

Your answers **must** reflect the output of your code for these functions. No credit will be given to answers if you have not submitted the respective correct function implementation in Scheme.

Question 11: What output do you get for the call:

```
(union '(green eggs and) '(ham))
```

Question 12: What output do you get for the call:

```
(intersect '(stewed tomatoes and macaroni) '(macaroni and cheese))
```

Troubleshooting

This lab requires an independent study of the Scheme language. You are encouraged to use any web tutorials and resources to learn Scheme. Given the size of the class, I will not be able to debug your code for you. Please do not send panicked emails requesting I fix your bug for you. Allow yourself plenty of time, and use patience, perseverance, and the internet to debug your code. I will gladly answer clarifying questions about the goals and instructions of the Lab assignment.

Lab Questions

The following questions are for feedback and evaluation purposes. Points are awarded for any sincere answer.

Question 13: Name something you like about Scheme. Explain.

Question 14: Name something you dislike about Scheme. Explain.

Question 15: Did you enjoy this lab? Which aspects did you like and/or dislike?

Question 16: Approximately how many hours did you spend on this lab?

Question 17: Do you think you will use Scheme again? For which type(s) of project(s)?

Extra Credit (10 pts)

Write a function `flatten` that takes two lists `lst1` and `lst2` and returns a new list that contains all elements in both lists, but eliminates any embedded lists. You may either retain or eliminate duplicates as your choice. You may freely use of any of the functions you wrote for this lab.

```
(define (flatten lst1 lst2)

   ; complete this function definition

)
```

Question EC1: Calling `flatten '(1 (2 3) 5) '(8 (13 (21 34) 55))`, what output do you get?

Question EC2: Describe, in a few English sentences, how your accomplished your list flattening.

Submission

Each student will complete and submit this assignment individually. Do not consult with others. However, you are encouraged to use the internet to learn Scheme **but not to research the questions asked in this lab**.

Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.

Save the final version of your program as `[lastname]_[firstname].lab3.scm`. Type your lab questions in plain text as `[lastname]_[firstname].lab3.txt`. Include your name in the text file.

Submit your files to the Lab3 dropbox folder on D2L. Do not archive your files but instead use two attachments. Submit your file before the due date as late submissions will not be accepted.