CSU2500 Exam 1 HONORS SUPPLEMENT – Fall 2011

Name: _____

Student Id (last 4 digits): _____

Instructor's Name: _____

- This supplement to Exam 1 is intended for students enrolled in the Honors section of 2500.
- See the instructions on the regular exam.

Problem	Points	/out of
1		/ 6
2		/ 10
3		/ 16
Total		/ 32

Good luck!

Problem 1 Here is a simple data definition for a list of symbols: 6 POINTS

```
;; An LoS is one of:
;; - empty
;; - (cons Symbol LoS)
```

Design the `remove-all` program which takes two lists of symbols and removes all occurrences symbols in the first list from the second one. For example:

```
(remove-all '(x q r) '(q x y x z x)) --> '(y z)
```

You may design helper functions as needed, but they should be designed according to the recipe.

[Here is some more space for the previous problem.]

Problem 2 For you first co-op, you've landed a lucrative job in San Francisco working for Twitter. On day 1, your boss, Steve Jenson (`@stevej`) calls you in and says there are all kinds of problems with Twitter's current design. He asks you to redevelop the core functionality underlying the twitterverse. Luckily the house style at Twitter follows the basic elements of the design recipe, so you're able to communicate fluently from the get-go. 10 POINTS

Steve explains the basic data definition for a message:

```
;; A Msg is a (make-msg User Content)
(define-struct msg (author content))

;; A Content is one of:
;; - empty
;; - (cons User Content)
;; - (cons String Content)

;; A User is a Symbol.
```

A message is a piece of compound data consisting of the message's author and its content. The content is a list of things that are either text (strings) or mentions of Twitter users (symbols).[1]

For example, this is a message:

```
(make-msg 'dickc
          '("NEU co-ops working with " stevej "!"))
```

What Steve wants is to compute the list of users mentioned in a message. In the example above, only `stevej` is mentioned, but multiple users may be mentioned:

```
(make-msg 'lambda_calculus
          '("Exam 1 mentions " RealmOfRacket ", "
            stevej ", and " dickc))
```

This message mentions `RealmOfRacket`, `stevej`, and `dickc`. Note that the author is not considered to be mentioned unless they refer to themselves explicitly in the message. So for example:

[1]For the purposes of this exam we completely ignore the 140 character limit on Twitter messages—in your spare time, you might think about how to add this feature.

```
(make-msg 'dickc '(dickc ": gr8est CEO eva!!"))
```

does mention `dickc`, while this one does not:

```
(make-msg 'dickc '(stevej ": gr8est hacker eva!!"))
```

Computing the mentioned list should be easy, but here's the twist: the list of user names should be *in the order they were first mentioned* and with *no duplicates*. Here are some examples:

```
(mentioned-msg (make-msg 'A '(A B C)))   --> '(A B C)
(mentioned-msg (make-msg 'A '(B B C B))) --> '(B C)
(mentioned-msg (make-msg 'A '(C C B C))) --> '(C B)
```

Design the function `mentioned-msg` that solves Steve's problem. Feel free to use any functions you've previously developed.

[Here is some more space for the previous problem.]

[Here is some more space for the previous problem.]

Problem 3 So Steve only told you part of Twitter's core data definitions. The real thing is slightly more complicated. An important feature for Twitter is that you can also "retweet" something that somebody else wrote and add your own comments to it. It's just like forwarding an email; you send out the original message and can also add some text of you own. As you know, email forward chains can get quite big, and the same is true of "retweet" chains. The data definition for a tweet is: 16 POINTS

```
;; A Tweet is one of:
;; - Msg
;; - (make-rt Msg Tweet)
(define-struct rt (comment tweet))
```

where `Msg` is defined as given above.

So if `stevej` retweeted `dickc`'s earlier message, it would look like this:

```
(make-rt
  (make-msg 'stevej '("They sure are!"))
  (make-msg 'dickc
            '("NEU co-ops working with " stevej "!")))
```

For simplicity, we will say that only `stevej` is mentioned in this tweet; it's easier if we just think of people as being mentioned if they are explicitly mentioned in a message. (We don't include retweeted authors.)

Design the `mentioned-tweet` function that consumes a tweet and produces a list of mentioned users in order and with no duplicates. For the ordering of a retweet, all mentions in the comment should come before mentions in the retweeted tweet.

[Here is some more space for the previous problem.]

[Here is some more space for the previous problem.]