CS 571
Quiz 3 Solution

Oct 10 **Closed book**
15 points **Closed notes**

Important Reminder: As per the course Academic Honesty Statement, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

Please ensure that you have filled-in BOTH your name and B-number in the bubbles on the provided grid-sheet.

For each of the following questions, select a **single** alternative on the grid-sheet.

There are 7 questions with 2-points per question; there is 1-point for submitting the quiz.

1. Which of the following languages over the vocabulary of square-brackets { `[`, `]` } is not expressible using standard regular expressions?

   (a) Strings of even length. Examples include the empty string, `][`, `[[[]` and `[[]]`.

   (b) Strings of even length which consist of balanced brackets. Examples include the empty string, `[]` and `[][[]]`.

   (c) Strings of even length containing 2-or-more `]`'s followed by 0-or-more `[`'s. Examples include `]]`, `]]][` and `]][[`.

   (d) Strings of length less-than-or-equal-to 4 which consist of balanced brackets. Examples include the empty string, `[]` and `[][]`.

   (e) Strings whose length is exactly 4. Examples include `]][[`, `[[[]` and `[][]`.

Answer: (b).

Regular expressions cannot describe arbitrary balanced constructs. Hence (b) is not expressible. (d) and (e) can be described by a regex which simply enumerates all possibilities. (a) can be described by the regex `( \[\[ | \[\] | \]\[ | \]\] )*` and (c) can be described by the regex `(\]\])+ (\]\[)? (\[\[)*`.

2. In Javascript, *hoisting* refers to:

   (a) Moving `let` declarations to the start of a block.

   (b) Moving `let` declarations to the start of a function.

   (c) Moving `var` declarations to the start of a block.

   (d) Moving `var` declarations to the start of a function.

   (e) Moving `var` declarations to the `window` object.

   Answer: (d).

   This is the standard definition of the term *hoisting* within the Javascript community. This behavior can surprise programmers coming to Javascript from a language where variables have block scope.

3. What should be the value of the following Scheme expression?

```
(length '( 1 (2 3) () () (()) ))
```

   (a) It will result in an error since the list is not a proper list.

   (b) 3

   (c) 4

   (d) 5

   (e) 6

   Answer: (d).

   The `length` function counts the number of top-level elements in the list and the list contains the following top-level elements: `1`, `(2 3)`, `'()`, `'()` and `'(())`. The last element is a 1-element list containing `'()`.

4. Which of the following is the most accurate characterization of the semantics of `cons`, `car` and `cdr` in Scheme?

   (a) `cons` constructs a list, `car` returns the head of the list, `cdr` returns the tail of the list.

   (b) `cons` constructs a list, `car` returns the tail of the list, `cdr` returns the head of the list.

   (c) `cons` constructs a pair, `car` returns the first element of the pair, `cdr` returns the second element of the pair.

   (d) `cons` constructs a list, `car` returns the first element of the list and `cdr` returns the second element of the list.

   (e) `cons` constructs a pair, `car` returns the second element of the pair, `cdr` returns the first element of the pair.

   Answer: (c).

   `cons` constructs a pair, not a list (which is constructed by `list`); `car` accesses the first element of the pair and `cdr` the second.

5. What should be the value of the following Scheme expression?

```
(caddr '(a b c d e))
```

   (a) `'b`.

   (b) `'c`.

   (c) `'d`.

   (d) `'(c d e)`

   (e) `'(d e)`

   Answer: (b).

   `cdr` is `'(b c d e)`; `cddr` is `'(c d e)`; hence `caddr` is `'c`.

6. What should be the value of the following Scheme expression?

```
(cdddr '(a b c d e))
```

(a) `'b`.

(b) `'c`.

(c) `'d`.

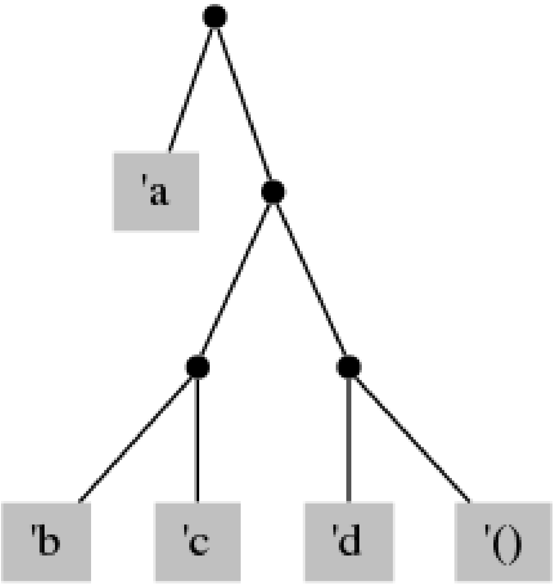
(d) `'(c d e)`

(e) `'(d e)`

Answer: (e).

`cdr` is `'(b c d e)`; `cddr` is `'(c d e)`; hence `cdddr` is `'(d e)`.

7. Given the following tree structure:



which of the following Scheme expressions best describes the structure?

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

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

(c) `'(a (b . c) d)`

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

(e) `'(a (b c) (d . ()))`

Answer: (c)

Since the right-spine of the figure ends with a `'()`, the figure represents a proper list. The first element is `'a`, the second element is an improper list `'(b . c)` (since it does not end with a `'()`) and the last element is `'d`.