CS 135 Winter 2018: Practice Questions

Note: these practice questions were prepared without seeing the midterm, and they do not cover everything that could be on the midterm. Completing only these questions is not sufficient preparation for the midterm.

1. The restaurant Racket King uses structures to represent information about its burgers. A burger is a structure of type Burger with four fields (in the following order):
   - patty, a symbol that is one of 'veggie', 'chicken, or 'beef.
   - veg-count, a natural number representing the number of vegetable toppings.
   - bun-type, which is one of 'white' or 'whole-wheat.
   - cheese?, which is true if the burger has cheese on it, and false otherwise.

To avoid overfilling an individual burger, the veg-count must be at most 10.

(a) Write a structure definition for burger and a data definition for Burger.
(b) Write a template burger-template that consumes a Burger. Remember to include a contract.
(c) According to Racket King, the following are considered healthy characteristics of a Burger:
   - The patty is either 'veggie or 'chicken.
   - The veg-count of a Burger is at least 5.
   - The bun-type is 'whole-wheat.
   - The Burger does not have cheese on it (that is, cheese? is false). Unfortunately, Racket King uses processed cheese on all of its burgers, which they do not consider to be healthy.

Write a function healthy?, that consumes a Burger, and produces true if that Burger has at least two of the four healthy characteristics mentioned above, and false otherwise. Avoid enumerating every single possibility that would make a burger healthy.

(d) Write a function health-score that consumes a list of Burgers, and produces the number of burgers from the consumed list that are considered healthy, according to the characteristics above. However, for every burger in the consumed list that has zero (0) healthy characteristics, the number produced is subtracted by one (1). For example, (health-score (list (make-burger 'veggie 4 'whole-wheat false) (make-burger 'beef 0 'white true))) would produce 0.

(e) Write a function add-veggie-count, that consumes a natural number, a symbol that is one of 'veggie, 'chicken, 'beef, or 'all, and a list of Burgers. The function produces a list of Burgers, where each burger is modified according to the following rules:
   - If the consumed symbol is 'all, then the veg-count field of every Burger in the consumed list will be incremented by the consumed natural number. Recall that a Burger cannot have a veg-count greater than 10, if incrementing the veg-count field were to exceed this threshold.
   - If the consumed symbol is 'veggie, 'chicken, or 'beef, only the Burgers in the consumed list that have the same patty field as the consumed symbol should have their veg-count field incremented (in the same way as described above).
   - All other Burgers in the consumed list are unaltered.

The order of the burgers in the consumed and produced lists should be the same.
2. Zainab is very picky about the characters in strings. As she searches through a string one character at a
time, starting from the first character of the string, she modifies that string according to the following
rules:

- Zainab has no issues with the empty string (""), and will not modify it in any way.
- Zainab loves seeing the letter Z appear in any string. Upon encountering a lower-case or
  upper-case Z character, the single occurrence of a Z will be replaced with two consecutive
  Z’s, both with the same case as the original Z.
- Zainab does not like the letter E, as she has a hard time writing them out. As soon as Zainab sees
  a single occurrence of a lower-case or upper-case e, it is removed from the string. Zainab will
  stop searching through the rest of the string at that point, and simply produce the rest of the
  characters in the string (even if they should have been removed or altered according to her rules).
- Zainab insists that all other upper-case characters are replaced with their corresponding
  lower-case characters. For example, the upper-case letter S would be replaced with the
  lower-case letter s in the string.
- Finally, all other characters in the string will remain unchanged according to Zainab.

Write a function Zainab-convert, that consumes a string, and produces the result of modifying that
string according to Zainab’s rules above. Note that it is appropriate to have a capital letter in the
function name, as Zainab is a proper noun. You may find the functions char-ci=?, char-upper-case?,
and char-downcase helpful.

3. Consider the following constant definition:
   (define question3 (cons "blue"
                        (cons (cons "green" (cons 57 (cons 38 empty)))
                              (cons (cons 'red
                                        (cons (cons 'question (cons 'three empty))
                                              (cons empty empty)))))))

   (a) Rewrite the definition of question3, using only the built-in list function to construct the list, and
       without using empty, cons, or quoted list notation.
   (b) Rewrite the definition of question3, using only quoted list notation.
   (c) State the value of (rest (first (rest (second question3)))), or explain why evaluating this
       expression would produce an error.
   (d) State the value of (length (cons (second question3) (fourth question3))), or explain why
       evaluating this expression would produce an error.

4. Write a function sum-odds that consumes two integers start and end, where start <= end, and
   produces the sum of all the odd integers in the range from start to end inclusive. For example,
   (sum-odds -4 7) produces 12.
5. Consider the following data definition:

;; A MysteryNat is one of 
;; 1 
;; 3 
;; (+ MysteryNat 4)

(a) Write a template mysterynat-template that consumes a MysteryNat.
(b) Write a function base-case-3? that consumes a MysteryNat and produces true if the consumed MysteryNat was recursively built up from the base case of 3 according to the data definition above, and false otherwise. The function should also produce true if the consumed MysteryNat is 3 itself. You may not use any form of multiplication or division. Your function should use pure structural recursion and follow from the template in part (a). Include a contract and examples.

6. Consider the following Racket function:

(define (question6 a b c)
  (cond
   [(odd? c) 'no]
   [(boolean? b) 'yes]
   [(< a 3) "Bob"]
   [(< a 9) "John"]
   [else (> = c 15)])
)

(a) Give the contract that most closely fits the question6 function, such that the types are as general as possible, but any function application following this contract will not produce an error.
(b) Give a complete set of tests that will sufficiently test this function. Express your answer as a sequence of check-expects. For each test, write a short comment that describes what part of the function is being covered by the test.

7. Consider the following function definition:

(define (question7 a b)
  (cond
   [(= (string-length a) (string-length b)) true]
   [(integer? (/ (string-length b) 2))
    (odd? (string-length a))]
   [(not (zero? (remainder (string-length b) 2)))
    (even? (string-length b))]
   [else false]])
)

(a) Write a purpose and contract for the function question7. Your purpose must describe what the function question7 does, and cannot simply be a direct English translation of the code above. Remember that the purpose should not describe or explain how the function works.
(b) Rewrite the definition of question7 using only Boolean expressions, and no instances of cond. Feel free to make any simplifications to the code above by using other mathematical functions and predicates, as you see appropriate.
8. Suppose that the following definitions have been fully processed in **Beginning Student with List Abbreviations**:

```
(define-struct review (time date location))

(define start-time 1630)
(define end-time (+ start-time 190))

(define cs135 (make-review end-time "Feb 14" 'MC))
(define cs135-lst (list start-time "Feb 14" 'MC))
```

```
(define (convert val)
  (cond
   [(cons? val) (make-review (first val) (rest val) (empty? cs135-lst))]
   [(review? val) (list (review-location val) (review-date cs135) (third val))]
   [else val])
)
```

```
(define (stepping a b c)
  (cond
   [(and (not (review? (make-review a b c))) c) "correct"]
   [(or (> a end-time) (> b start-time)) (convert cs135)]
   [else "incorrect"]))
)
```

```
(define (recurse lst)
  (cond
   [(empty? lst) 5]
   [else (+ (string-length (stepping (first lst) 4 true)) (recurse (rest lst)))]))
)
```

Using the rules described in class and the course notes, provide the first six substitution steps of evaluating the following Racket expressions, in addition to the final value that would result. If the evaluation would produce an error, describe the error. You may choose to use **cons**, **list**, or quoted-list notation as appropriate throughout your evaluation. However, switching between these notations does not count as a substitution step. (For additional practice, feel free to provide the full trace.)

(a) (convert cs135-lst)
(b) (stepping end-time 1730 false)
(c) (convert cs135)
(d) (recurse (list 8))