Instructions:

1. This is a closed book, closed notes exam. You are allowed to use one 8.5 x 11” sheet of paper with writing or printing on both sides during the exam.

2. State your assumptions. Partial credit will be given. Grading will be based on completeness, correctness, clarity, and neatness.

3. If you need extra paper, ask your professor or proctor for a blank sheet of paper. Clearly label your answers on the extra paper. If you use any extra paper, write your name on the extra pages and hand them in with your exam.

4. You have 75 minutes to complete this exam. Relax and BREATHE.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Short answer about statements using logical operators</td>
<td>3</td>
</tr>
<tr>
<td>2. Creating and accessing lists</td>
<td>10</td>
</tr>
<tr>
<td>3. Define statements and the Global Environment</td>
<td>10</td>
</tr>
<tr>
<td>4. Writing a list-processing function</td>
<td>10</td>
</tr>
<tr>
<td>5. Tracing the execution of a mystery function</td>
<td>8</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>40</strong></td>
</tr>
</tbody>
</table>
1. (3 points) Answer the following questions about the evaluation of logical expressions.

   i. Suppose \( a, b, c \) and \( d \) are defined as \#t. What would be the result of evaluating the following expression (true or false)?

      \[(\text{and} \ (\text{not} \ a) \ (\text{or} \ b \ c \ d))\]

   ii. Suppose \( a, b, c \) and \( d \) are defined as \#f. What would be the result of evaluating the following expressions (true or false)?

      \[(\text{or} \ a \ (\text{not} \ (\text{not} \ b)) \ (\text{not} \ (\text{and} \ c \ d)))\]

   iii. Circle any parts of the two expressions above that are NOT evaluated with this set of inputs. If you think no parts are unevaluated, write ”all evaluated.”
2. (10 points) Consider the following 3 example lists: EVENS, ODDS, and THREES.

;;; Examples
(define EVENS (cons 2 (cons 4 (cons 6 (cons 8 (cons 10 empty))))))
(define ODDS (cons 1 (cons 3 (cons 5 (cons 7 (cons 9 empty))))))
(define THREES (cons 3 (cons 6 (cons 9 (cons 12 (cons 15 empty))))))

(a) (4 points) Show the result of evaluating the following expressions, keeping in mind that the evaluation may result in an error. If you think the statement will cause an error, just write ERROR for the result of evaluation.

(first (rest THREES)) =>
(rest (rest (first ODDS))) =>
(append ODDS (list 11)) =>
(rest (rest (rest (rest EVENS)))) =>

(b) (6 points) Write expressions using only first, rest, cons, and/or empty on the lists EVENS, ODDS, and THREES, defined above to produce the lists or list elements specified below.

– Write an expression that evaluates to the list containing the last three numbers from the ODDS list.

– Write an expression that evaluates to the list containing the first number of each list (i.e., the list containing 2, 1, 3).

– Write an expression that evaluates to the list containing the first element of EVENS, the second element of ODDS, and the third element of THREES (i.e., the list containing 2, 3, 9).
3. (10 points) Consider the define statements and representation of the Global Environment given below:

```
(define A 3)
(define B 2)
(define C (* A B))
(define X (+ C 3))
(define (fun X Y) (* X (+ Y 4)))
```

(a) (5 points) Show, in the table below, the additions to the global environment created by the 5 definitions above when they are executed in DrRacket (assume entries are added from the top of the table downward).

<table>
<thead>
<tr>
<th>Global Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

(b) (5 points) Using the Global Environment from part (a), show the result of evaluating each of the following expressions, keeping in mind that the evaluation may cause an error message (the > is the Interactions Window prompt). If you think the expression causes an error, just write ERROR.

```
> (expt C 2)

> (* B 1000)

> (string-append A B)

> (string? X)

> (fun A X)
```
4. (10 points) Write a function called `swapper`. This function should consume two strings, `str1` and `str2`, and a list of strings called `los`. The function should return a list of strings in which every instance of `str1` has been replaced by `str2` and every instance of `str2` has been replaced by `str1` in `los`. Write this function in 2 versions, one version using regular recursion and the other version using an accumulator.

```
; Contract: (swapper str1 str2 los) -> list of strings
; Input: str1,str2:strings; los:list of strings
; Purpose: Replace str1 with str2 and str2 with str1 everywhere
; they appear in los

; Pre-function tests:
(check-expect (swapper "a" "b" (list "a" "b" "a" "b" "b"))
  (list "b" "a" "b" "a" "a"))
(check-expect (swapper "a" "b" empty) empty)
(check-expect (swapper "cat" "dog" (list "my" "dog" "has" "cat" "fleas"))
  (list "my" "cat" "has" "dog" "fleas"))

; Function definition:
(define (swapper str1 str2 los)

```

5. (8 points) Parts (a) and (b) refer to the `mystery?` function:

```lisp
(define (mystery? x)
  (or (empty? (rest x))
      (and (> (first (rest x)) (first x))
           (mystery? (rest x))))
```

(a) (6 points) Show what the function `mystery?` returns for the following inputs or indicate that the function call would return an error by writing ERROR:

- `(mystery? (list 2 5 9 17))` .................________________
- `(mystery? (list 1 2 5 4))` .................________________
- `(mystery? (list 8))` .......................________________
- `(mystery? (list 8 8 8 8))` .................________________
- `(mystery? empty)` ..........................________________
- `(mystery? (list 1 2 2 4))` .................________________

(b) (2 points) Clearly, the `mystery?` function consumes a flat list of numbers. But what else must we assume about the list `mystery?` consumes as input to ensure calling the function with that list as an argument does not produce an error? If you think the function works for any size flat list of numbers, just write “no further assumptions are necessary.”
6. (6 points) Answer the following questions about lists.

(a) (2 points) Use cons and/or empty to construct the equivalent of the following lists:

(list 0 1 2 3 4 5)

(list)

(b) (2 points) Use list to construct the equivalent of the following lists:

(cons 5 (cons 4 (cons 3 (cons 2 (cons 1 empty))))))

(cons empty empty)

(c) (2 points) Write the lists returned by the following expressions, using the list function in your answers:

(list (symbol=? 'a 'b) (symbol=? 'c 'c) false)

(list (+ 10 20) (* 10 20) (/ 10 20))

(d) (2 points) Write the lists returned by the following expressions, using the apostrophe notation and NOT the list or cons functions in your answers:
7. (5 points) Give the result of calling the following function with input (secret 4)

(define (secret x)
  (cond
   [(= x 0)
    (begin
     (printf "\nbase case, x = ~a\n" x)
     0))
   [(= x 1)
    (begin
     (printf "\nbase case, x = ~a\n" x)
     1))
   [else
    (begin
     (printf "\nrecursive case, x = ~a\n" x)
     (+ (secret (- x 2)) (secret (- x 1))))]))}