Name: _______________________________ Signature: _______________________________

With my signature, I swear that I did not already and will not discuss the contents of this exam with anyone until after my graded exam is returned.

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 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 for it. Clearly label your answers on extra sheets if you use any, write your name on each one, and hand them in with your exam.

4. There are 6 questions, worth a total of 55 points, on 4 pages (including this page), written front and back.

5. You have 90 minutes to complete this exam. Relax and breathe.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>received</th>
<th>possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Short answers, evaluating, and writing expressions</td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2. Define, the Global Environment, and evaluating expressions</td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>3. Write a function that contains decision statements</td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>4. Short answer about special forms and built-in functions</td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>5. Rewriting and adding comments to a recursive function</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>6. Recognizing errors in a recursive function</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>55</td>
</tr>
</tbody>
</table>
1. (10 points)

(a) (2 points) Define the terms *syntax* and *semantics* in regard to a language.

(b) (2 points) Suppose a and b are defined as #f, and c and d are defined as #t. What would be the result of evaluating the following expressions (true or false)?

i. \( \text{(and (not a) (or b c d))} \)

ii. \( \text{(or a (not (not b)) (not (and c d)))} \)

(c) (6 points) Use the method shown in class to convert the following infix expressions to prefix notation. *Just convert the expressions to prefix notation, do not simplify or change the order of operands:*

i. \( (x - 32) \times (\frac{5}{3}) \)

ii. \( (x + 3 \times y) \times (x - y) \)

iii. \( a + (b^3 - d)/e \times f \)  
    *Hint: you may use expt*
2. (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 (lambda (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>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<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):

> (expt C 2)

> (* B 1000)

> (string-append A B)

> (string? X)

> (fun A X)
3. (15 points) Define a function, called `name-card`, that consumes an integer in the range 1...13, and a quoted symbol 'H, 'D, 'C, or 'S. The function should use the integer input to calculate the output depicted in the table below:

<table>
<thead>
<tr>
<th>First Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Ace&quot;</td>
</tr>
<tr>
<td>2...10</td>
<td>convert given number to a string</td>
</tr>
<tr>
<td>11</td>
<td>&quot;Jack&quot;</td>
</tr>
<tr>
<td>12</td>
<td>&quot;Queen&quot;</td>
</tr>
<tr>
<td>13</td>
<td>&quot;King&quot;</td>
</tr>
</tbody>
</table>

Your function should append one of the string outputs shown in the table above to the string generated for the quoted symbol parameter: " of Hearts" (for input 'H), " of Diamonds" (for input 'D), " of Clubs" (for input 'C), and " of Spades" (for input 'S). Read the 9 calls to this function in the interactions window output below to better understand how this function should work (the > is the interactions window prompt).

> (name-card 11 'D) > (name-card 10 'H) > (name-card 4 'C)
"Jack of Diamonds" "10 of Hearts" "4 of Clubs"
> (name-card 1 'H) > (name-card 13 'D) > (name-card 1 'S)
"Ace of Hearts" "King of Diamonds" "Ace of Spades"
> (name-card 3 'C) > (name-card 12 'S) > (name-card 6 'H)
"3 of Clubs" "Queen of Spades" "6 of Hearts"

On the lines indicated below, write a contract, header, (brief) purpose and at least 3 check-expect statements to demonstrate your understanding of how the function should work. Write the function definition for `name-card` on the facing page (page 5). Choose descriptive parameter names for the input parameters (e.g., single letter parameter names are not descriptive of what the parameters represent). There is no need to define constants for the literal values you use in the function and you do not have to account for erroneous input.

_Built-in functions that may be useful in your function:_ `number->string`, `symbol=?`, and `string-append`.

; Contract:

; Header:

; Purpose:

; Pre-function tests:
(Solution space for problem 3).

; Function definition:
4. (10 points) Given the following character sequences (words) (a through e below), indicate whether the word is a special form, a built-in function, or neither. If you answer special form, indicate how the form is used and give the return type (or void if a side-effect is the only thing returned).

For example:

```plaintext
if    SPECIAL FORM used to make decisions. if has 3 arguments. The first argument is a boolean expression, bool. If bool is true, the second argument is evaluated, otherwise the third argument is evaluated. The value of either the second or the third argument is returned.
```

If you answer built-in function, give an example of calling the function on valid arguments and indicate the output of the function call. For example:

```plaintext
+    A built-in function that takes numeric arguments and returns a number, the sum of its arguments.
Example: (+ 5 7 4 8 2) ==> 26
```

If you think the given word is an undefined name, write “neither”. For example:

```plaintext
my-func    neither
```

(a) boolean?

(b) λ

(c) >=

(d) or

(e) string+
5. (5 points) This question involves a recursive function.

(a) (1 point) Suppose you were asked to write a regular recursive function that produces the sum of the cube of each number between 0 and n (no negative numbers allowed). This sum could be expressed as

\[ 0^3 + 1^3 + 2^3 + 3^3 + \ldots + n^3 \]

Would the following function compute this sum? Answer “yes” or “no”.

\[
\text{(define sum-cube}
\text{ (lambda (n))}
\text{ (cond}
\text{ [ (= n 1) 0]}
\text{ [else}
\text{ (+ (* n n) (sum-cube (- n 2)))])])
\]

(b) (4 points) Rewrite the sum-cube function below, correcting any errors in the function (if you found any errors), and adding appropriate comments on each clause.
6. (5 points) Consider the count-down function, given in the space below on the left. Assume the line numbers and the text to the right are not part of the function.

The count-down function is supposed to produce the output shown on the right when run in the interactions window. Unfortunately, in its current state, it would fail to do so.

1. (define count-down > (count-down 5)
2.  (lambda (n) 5
3.   (cond 4
4.     [(n = 0) 3
5.       (begin 2
6.         (printf "~a-%% n) 1
7.         (printf "BLAST-OFF-%")]) 0
8.       [else 3
9.         (begin >
10.           (printf "~a-%% n) 1
11.           (count-down (n - 1)))]))

(b) (3 points) Write a contract, header, and purpose for the count-down function.

(a) (2 points) Indicate which lines in this function would need to be changed in order for the function to produce the desired output when run in the interactions window. After writing each line number that needs to be changed, re-write the line in its correct form.