Name: _____________________    Student #: _________________
Signature: _____________________

Lab Section: _____
Lecture Section (circle one):      Gregor       Kurt

Important notes about this examination
1. You have 90 minutes to write this examination.
2. This exam will be graded significantly on how well you follow the design recipes. You have been given a copy of the Recipe Summary and Template Rules. Use them!
3. Put away your books, notebooks, cell phones... everything but pens, pencils, erasers and this exam.
4. There are blank pages at the end of the exam you can use for extra space. When you need extra space, mark that clearly both at the original problem and on the extra sheet.

Rules Governing Formal Examinations
1. Each candidate must be prepared to produce, upon request, a UBCcard for identification.
2. Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities in examination questions.
3. No candidate shall be permitted to enter the examination room after the expiration of one-half hour from the scheduled starting time, or to leave during the first half hour of the examination.
4. Candidates suspected of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action:
   - having at the place of writing any books, papers or memoranda, calculators, computers, sound or image players/recorders/transmitters (including telephones), or other memory aid devices, other than those authorized by the examiners;
   - speaking or communicating with other candidates; and
   - purposely exposing written papers to the view of other candidates or imaging devices. The plea of accident or forgetfulness shall not be received.
5. Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.
6. Candidates must follow any additional examination rules or directions communicated by the instructor or invigilator.

Please do not write in this space:

<table>
<thead>
<tr>
<th>Question</th>
<th>Mark</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 - (24 points)

The following program has a function definition followed by a marked expression.

(define (f n)
  (local [(define (g m)
            [cond [(= m n) m]
                  [else (* m (g (add1 m)))]])]
    (g 1)))

(f 2) ;<<< MARKED EXPRESSION

In this problem you need to show how the evaluation of the marked expression proceeds over time. You must show the state of the evaluation at three points in time. You may use the scratch sheets if you need to, but be sure to show the entire current state of the execution at each point in time, and be sure not to include extraneous expressions in that state. Be sure to show any lifted definitions at the appropriate points.

(A) Right after the call to \texttt{f} has been replaced by the appropriately substituted body of \texttt{f}. In other words, right when the evaluation of the \texttt{local} is about to begin.

(B) Right after renaming and lifting, in other words right when the body of the \texttt{local} is about to be evaluated.
(C) Right when the evaluation of the `cond` expression is about to begin.
Problem 2 - (50 points)

Consider the following partial data definition:

```
(define-struct node (k v l r))
;;; BinarySearchTree is one of:
;;;   - false
;;;   - (make-node Integer String BinarySearchTree BinarySearchTree)
;;; interp. a binary search tree, each node has a key and a value
;;; Invariant - for every node:
;;;   - every k in the left branch has a k less than the node’s
;;;   - every k in the right branch has a k greater than the node’s
```

A) Neatly annotate the type comments above by drawing a line from each type reference to the corresponding type definition. All your arrows should have the 'pointy-end' at one of the names that appear right before 'is'. Neatly label each line with one of MR, SR or R, depending on whether the reference is a mutual reference, self reference or an ordinary reference.

B) Write THREE examples for BinarySearchTree. The first should be a base case, the third should follow the at least two wide and two deep rule.
(C) Write the template for a function operating on BinarySearchTree.

(D) Design a function that inserts a new key and value into a BinarySearchTree. You may assume that the tree does not already contain a key with that value. You do NOT need to rebalance the tree after insertion, just insert the new key and value in a place in the tree that preserves the invariant described in the data definition.
Problem 3 - (26 points)

Simplify the definition of each of the following functions using either map or filter. The signatures of map and filter are:

map:   (X -> Y) (listof X) -> (listof Y)
filter: (X -> Boolean) (listof X) -> (listof X)

(A)
(define (bar c lod)
  (cond [(empty? lod) empty]
        [else
         (if (string=? (dot-color (first lod)) c)
             (cons (first lod) (bar c (rest lod)))
             (bar c (rest lod)))]))

(B)
(define (foo lon)
  (cond [(empty? lon) empty]
        [else
         (cons (sqr (first lon))
              (foo (rest lon)))]))