CS351: Exam #2

April 15, 1999

NAME:____

1. (20 points) Below is a buggy definition of a macro `mcase` that branches based on the value of a test (similar to the Common Lisp `case`). Note ‘ is backquote ’ is quote.

```
(defmacro mcase (key &rest clauses)
  `(let ((value , key))
     (cond , (mapcar #'(lambda (clause)
                          `((eq value ,'(first clause)) ,(rest clause)))
                     clauses))))
```

The form

```
(mcase (first l)
    (A  (reverse l))
    (B  (rest l)))
```

should expand into the form

```
(let ((value (first l)))
    (cond ((eq value 'A) (reverse l))
          ((eq value 'B) (rest l))))
```

What does it actually expand into?

Show the corrections required to fix the macro. Hint: three small changes are needed.

2. (20 points) Show exactly what happens when each of the following expressions is evaluated:

```
(defvar *x* '(10 9 8))
(defun foo1 (*x* y)
   (if (null y)
       (foo2)
       (foo1 (rest *x*) (rest y))))
(defun foo2 ()
     (first *x*))
(foo1 '(a b c) '(1 2))
=>
```

```
*x*
=>
```

```
(setf x '(a b c)) (setf y '(1 2 3))
(setf z (append x y))
(setf (rest (member 2 z)) '(a d))
(delete 'a z)
=>
```

```
x
=>
```

```
y
=>
```

```
z
=>
```

3. (20 points) Below is a buggy definition of a destructive set union function. It is supposed to take two lists representing sets and return their union. It can destructively modify its first argument but not its second. It should not do any consing. Assume the input sets do not contain duplicates.

```
(defun d-union (a b &key (test #'eql))
  "Adds elements of A to B unless they are already elements of B.
   Avoids consing by destructively modifying A but not B."
  (let ((result b))
    (loop (if (null a) (return result))
          ;; if first item in remainder of A is not a member of B,
          ;; then destructively add it to the front of result
          (unless (member (first a) b :test test)
            (setf (rest a) result)
            (setf result a))
          ;; go on to next element of A
          (setf a (rest a)))))
```

What is returned by `(d-union '(a b c) '(b e d))`?

Fix the bug. At most, you only need to add one line and change two existing lines.

4. (20 points) For the following pairs of patterns, show the binding list returned by `unify`, *and* the result of applying this binding list to the patterns (the result returned by `unifier`), or state why they do not unify (be specific and show the current binding list at the point the unification fails).

```
(member ?x (a b c)) and
(member ?y (?y . ?z))
```

```
(P ?x ?y (f ?x ?y)) and
(P (g ?w) ?z ?w))
```

```
(P ?x ?y (f ?x ?y)) and
(P (g ?w) ?v (f ?z ?z))
```

5. (20 points) Given the following logic program, show the answer(s) if any (and in the correct order if there is more then one) returned by the given query.

```
(<- (P ?x ?y) (R ?x ?y))
(<- (P ?x ?y) (R ?x ?z) (P ?z ?y))
(<- (R A B))
(<- (R A C))
(<- (R D A))
(<- (R E A))
(<- (R C F))

(?- (P ?x F))
```