COMP 280 : Assignment 4, Sample Solutions

1.
```
#|
A boolsymL is either:
- empty
- (cons boolsym boosymL)
An Answer is either:
- a boolean
- a boolsymL
|#
;; check-equiv-vars : formula formula vars -> Answer
;; returns true iff the two formulas agree on their values for every
;; combination of values on the variables in vars. We assume that
;; vars contains all variables appearing in either input formula.
;; Returns a list of boolean symbols representing the counterexample
;; if the two formulas are not equivalent.
(define (check-equiv-vars F1 F2 vars)
  (local ((define (check-equiv-vars-acc F1 F2 vars acc)
            (cond [(empty? vars) (if (equal? (eval F1) (eval F2))
                                     #t
                                     acc)]
                  [else (let ((answer_true (check-equiv-vars-acc (subst 'true (first vars) F1)
                                                                 (subst 'true (first vars) F2)
                                                                 (rest vars)
                                                                 (append acc (list 'true))))
                              (answer_false (check-equiv-vars-acc (subst 'false (first vars) F1)
                                                                  (subst 'false (first vars) F2)
                                                                  (rest vars)
                                                                  (append acc (list 'false)))))
                          (cond
                            [(and (boolean? answer_true) (boolean? answer_false)) #t]
                            [(list? answer_true) answer_true]
                            [(list? answer_false) answer_false]))])))
    (check-equiv-vars-acc F1 F2 vars empty)))
;; eval : formula -> boolean
;; reduces a formula containing only true and false to a boolean value
;;
;; subst : boolsym sym formula -> formula
;; replaces every occurrence of sym in formula with boolsym
```

2. (a) ∃ student : in-class(student) ∧ ¬ like(student, owls).
   There is a student in this class who dislikes owls.

   (b) ∀ student : in-class(student) → has-been(student, Sammy's).
   All students in this class have been to Sammy's.

   (c) ∀ student : in-class(student) → ∃ course : math(course) ∧ Rice(course) ∧¬ taken(student, course).
   Every student in this class has not taken at least one Rice mathematics course.

   (d) ∃ student : in-class(student) ∧ ∃ building : on-campus(building) ∧ ∀ room : in-building(room, building) → ¬ has-been(student, room).
   There is a student in this class who has not visited any room of some building on campus.

3. (a) Model: let student and prof be natural numbers n and m, and class-from(n, m) be $(n * m) = 0$.
   Then the statement becomes:

$\forall n\, \exists m : (n * m) = 0 \rightarrow \exists m\, \forall n : (n * m) = 0$

which is true, since both sides are true (pick zero for m).

Countermodel: let student and prof be natural numbers n and m, and class-from be equality. Then the statement becomes:

$\forall n\, \exists m : n = m \rightarrow \exists m\, \forall n : n = m$

which is false, since the left part of the statement is true (take m to be n) and the right part is false (or else all the natural numbers would have to be equal).

(b) Model: let bar be a natural number n, ball be zero, baz(m) be $(m + 0)$, and foo(n, m) be $n \geq m$. Then the statement becomes:

$\forall n : n \geq (0 + 0) \rightarrow \forall n : n \geq 0$

Countermodel: let bar be a natural number n, ball be 17, baz(m) be $(m - 17)$, and foo(n, m) be $n \geq m$. Then the statement becomes:

$\forall n : n \geq (17 - 17) \rightarrow \forall n : n \geq 17$

which is obviously false.

Extra credit: see the model given above.

4. (a) "$\forall x\, \forall y$" means "for all x and for all y", which is the same as "for all y and for all x".

   (b) "$\forall x\, \exists y$" means "for all x there is at least one y such that ... ", but the y can depend on the value of x, and be different each time.

   On the other hand, "$\exists y\, \forall x$" means "there is at least one y such that for all x ... ". So the same y works for all possible values of x.

   So the two parts of the statement do not have the same meaning.

5. If I be a model for $\exists x : (P(x) \vee Q(x))$ with domain D, then $\exists x : (P(x) \vee Q(x))$ is true for I, which means that $(P(d) \vee Q(d))$ is true for some $d \in D$. Therefore, at least one of $P(d)$ and $Q(d)$ is true for some $d \in D$. So at least one of $\exists x : P(x)$ and $\exists x : Q(x)$ is true for I. So $\exists x : P(x) \vee \exists x : Q(x)$ is true for I and I is a model for $\exists x : P(x) \vee \exists x : Q(x)$.

   If I is a model for $\exists x : P(x) \vee \exists x : Q(x)$ with domain D, then $\exists x : P(x) \vee \exists x : Q(x)$ is true for I, which means that at least one of $\exists x : P(x)$ and $\exists x : Q(x)$ is true for I, which means that there is some $d \in D$ such that at least one of $P(d)$ and $Q(d)$ is true. So there is some $d \in D$ such that $P(d) \vee Q(d)$ is true. So $\exists x : P(x) \vee Q(x)$ is true for I, and I is a model for $\exists x : P(x) \vee Q(x)$.

6. The statement is false. Consider the interpretation where the domain is $\{true, false\}$, $P(x)$ is x and $Q(x)$ is $\neg x$. Then the statement becomes:

   $\exists x : (x \wedge \neg x) \equiv \exists x : x \wedge \exists x : \neg x$.

   The left hand side of the statement is false, while the right hand side is true.

7. To negate an expression with n quantifiers, we can replace each $\exists$ in the sequence with $\forall$, each $\forall$ in the sequence with $\exists$, and negate the predicate inside the sequence.

   We will prove this statement by induction on the number of quantifiers in the sequence.

   Base case: there is a single quantifier, and the statement is therefore true (proved in class).

   Induction step: assume the statement is true for n. We now have to prove it for $n + 1$.

   Let $\bigotimes$ represent either $\exists$ or $\forall$, and $\overline{\bigotimes}$ represent $\bigotimes$ replaced with the other quantifier. Then:

   $\neg(\bigotimes x_1 \ldots \bigotimes x_{n+1} : P(x_1, \ldots, x_{n+1})) \equiv \neg(\bigotimes x_1 : Q(x_1)) \equiv \overline{\bigotimes} x_1 : \neg Q(x_1)$

   But $Q(x_1)$ is an expression with n quantifiers, so, by the induction hypothesis, $\neg Q(x_1) \equiv \overline{\bigotimes} x_2 \ldots \overline{\bigotimes} x_{n+1} : \neg P(x_1, \ldots, x_{n+1})$. So:

   $\neg(\bigotimes x_1 \ldots \bigotimes x_{n+1} : P(x_1, \ldots, x_{n+1})) \equiv \overline{\bigotimes} x_1 \overline{\bigotimes} x_2 \ldots \overline{\bigotimes} x_{n+1} : \neg P(x_1, \ldots, x_{n+1})$

   which proves the statement to be true for $n + 1$, which completes the proof.