Homework #1

due: 2013/09/24 P.M. 12:59

1. Formulate the arguments symbolically and determine whether each is valid.

Let

p: I miss the class more than 3 times.

q: I lose points.

r: I make a mistake on the first homework.

1.1

I miss the class more than 3 times if and only if I lose points.

I don't make a mistake and miss the class more than 3 times.

∴ I lose points.

$p \leftrightarrow q$

$\overline{r} \wedge \overline{p}$

$\therefore q$

This is invalid.

(I made a mistake on this problem. I missed 'I' after 'and' in second line. This is not the problem about English, so I decided below one is also correct.)

$p \leftrightarrow q$

$\overline{r} \wedge p$

$\therefore q$

This is valid.

1.2

If I miss the class more than 3 times, then I lose points.

If I don't make a mistake on the first homework, then I lose points.

I make a mistake on the first homework and I lose points.

∴ I don't miss the class more than 3 times.

$p \rightarrow q$

$\overline{r} \rightarrow q$

$r \wedge q$

$\therefore \overline{p}$

This is not valid.

2. Write truth table for the proposition

$P: (\overline{p} \leftrightarrow q) \wedge (r \vee \overline{q})$, and express it in disjunctive normal form.

truth table

p	q	r	P
T	T	T	F
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	F

disjunctive normal form

$$P = (p \wedge \overline{q} \wedge r) \vee (p \wedge \overline{q} \wedge \overline{r}) \vee (\overline{p} \wedge q \wedge r)$$
$$= p\overline{q}r + p\overline{q}\,\overline{r} + \overline{p}qr$$
$$= p\overline{q}(r + \overline{r}) + \overline{p}qr$$
$$= p\overline{q} + \overline{p}qr$$

3. Suppose a triangle which 3 side lengths are a, b and c.
Find the quantifier statement expressing condition of triangle and program it with **while** statement. (c.f. TP)

$$T(a,b,c) = a + b > c,\ when\, a \le b \le c \text{ and } a, b, c \in R^+$$
$$\forall a \forall b \exists c\, T(a,b,c)$$
$$\text{Let } A = B = C = R^+$$

```
function "∀a∈A∀b∈B∃c∈C: T(a, b, c)" ∈ B
(a, ∀1) := (first(A), F);
while ¬∀1 do
   (b, ∀2) := (first(B), F);
   while ¬∀2 do
      (c, ∃) := (first(C), F);
      while ¬∃ do
         if T(a, b, c) → ∃ := T | ¬T(a, b, c) → (c, ∃) := next²(C) fi
      od;
      if ∃ → (b, ∀2) := next²(B) | ¬∃ → return F fi
   od;
   (a, ∀1) := next²(A);
od;
return T;
```

(This is just my solution. I graded flexibly according to your definition.)