Deduction Exercises

Here are some theorems to prove using each of the different Boolean computational techniques. Prove that each sentence is a tautology.

Double negation: `(not (not A)) = A`

Modus Ponens: `(A and (A -> B)) -> B`

Subsumption: `(A and (A or B)) = A`

Resolution: `((A -> B) and ((not A) -> C)) -> (B or C)`

Drunken Liars/Fruit:

```
((L -> ((not G) -> A)) and
 (A -> ((D or (not C)) -> G)) and
 (not (H -> G))) -> (not (L and D))
```

To make the deduction exercises more relevant, here are some pseudocode fragments (using infix notation). Capital letters represent calls to existing subprograms. You should transform each into a more efficient code fragment using any of the deductive tools we have discussed.

1. `(not ((not A) or (not B)))`

2. `(if ((if test then A else True) and test) then A else True)`

3. `((not A) and (B or (not A)))`

4. `(block (if test1 then A else B); (if test2 then B else A); A)`

5.
```
(if (test1[A D E F] and (not test2[A B C D]))
 then ((not A) or (not E)) else True)

  test1[A D E F] =def= (if D then (A or (not (E or (not F)))) else True)
  test2[A B C D] =def= ((not (A or (not B))) and
                        (if C then (A or D) else True)
```

6.
```
(if test then A else B)

  A[] =def= (if test then C else True)
  B[] =def= (if test then D else True)
```

7. You are working on a Y2K problem, and you know that

```
  (if ((X has-two-digits) or (X isa-name-of-a-month))
   then (X isa-two-digit-month))
and
  (if ((X isa-name-of-a-month) or (not (X isa-two-digit-month)))
   then (if (not (X has-three-digits)) then (X isa-date)))
and
  (not ((X isa-two-digit-month) or (X has-three-digits)))

Prove that (X isa-date)
```

Determine which of these sentences is a tautology. For those that are not tautologies, determine a set of values for the relevant variables that satisfy the sentence.

1. `(or (if t t t) (if f f f))`

2. `(iff (and t f) (or t f))`

3. `(if (or a b) (or b a))`

4. `(if (not a) (or a a))`

5. `(and p p)`

6. `(and (or a b) (or (nor a b) (nor a b)))`

7. `(iff (if a a a) a)`

8. `(iff (or p (and q r)) (and (or p q) (or p r)))`

9. `(not (iff (not (and a (or a b))) (not a)))`

10. `(not (if (and a b) a b))`

11. `(if (if a b) (if a (or b c)))`

12. `(if (and p q) (if (and p q) p))`

13. `(or (and (or a b) (or a c)) (and (or a b) (or b e)))`

14. `(if (and (and p (not p)) (if r u)) (or q s) )`

15. `(if (and (if d (or a c)) (and d (not a))) c)`

16. `(if (and (if a b) (if b c) (if c d) (not d)) (not a))`

17. `(iff (if (if a b c) d e) (if a (if b d e) (if c d e)))`

18.
```
(and (or (not a) (not c)) (or (not a) c)
      (or a b) (or a c) (or a (not b) (not c)))
```

19.
```
(if (and (if (or a b) (or c d)) (if (or c f) h)
          (and e (not d)) (if e a)) (or h i) )
```

20.
```
(iff (nor (and a c) (and b (or a (not c)) (or c (not a))))
      (or (nor a (nor c (not b))) (nor c (nor a (not b)))))
```

21.
```
(and (or a c) (or (not a) (not c))
      (or (and (not a) (not b)) (and a b (not c)))
      (or (and (not b) (not c)) (and b c (not a))))
```

22.
```
(if (and (if (or b c) a) (if a (or s u))
          (and b (not s)) (if (and u a) (if w s))) (not w))
```

23.
```
(iff (or (and a b c) (and a (not b) c)
      (and a (not b) (not c)) (and (not a) (not b) c)
      (and (not a) (not b) (not c)))
   (or (not b) (and a c)))
```