```
--------------------------------------------------------------------------------------------------
Logic in Computer Science, April 20th, 2023. Time: 1h45min. No books or lecture notes
allowed.
--------------------------------------------------------------------------------------------------
 - Insert your answers on the dotted lines ... below, and only there.
 - When finished, upload this file with the same name: exam.txt
 - Use the text symbols:     &     v    -       ->         |=        A       E
     for                   AND   OR   NOT   IMPLIES   "SATISFIES"  FORALL  EXISTS  etc.,
like in:
    I |=  p & (q v -r)     (the interpretation I satisfies the formula p & (q v -r)  ).
   You can write  not (I |= F)  to express "I does not satisfy F",  or
                  not (F |= G)  to express "G is not a logical consequence of F"
   Also you can use subindices with "_". For example write x_i to denote x-sub-i.
--------------------------------------------------------------------------------------------------


Problem 1. (2.5 points).

1a) Let F, G, H be formulas. Prove that if F v G |= H then F & -H is unsatisfiable,
    using only the definition of propositional logic.

Answer:
        F v G |= H                                            ==>   [by def. of logical
consequence]
    AI, if I |= F v G then I |= H                             ==>   [by def. of |=]
    AI, if eval_I(F v G) = 1 then eval_I (H) = 1              ==>   [by def. of eval_I(v)]
    AI, if max(eval_I(F), eval_I(G)) = 1 then eval_I(H) = 1   ==>   [by def of max]
    AI, if eval_I(F) = 1 then eval_I_(H) = 1                  ==>   [by arithmetic]
    AI, if eval_I(F) = 1 then 1-eval_I(H) = 0                 ==>   [by def. of eval_I(-)]
    AI, if eval_I(F) = 1 then eval_I(-H) = 0                  ==>   [by def. of min]
    AI, min(eval_I(F), eval_I(-H)) = 0                        ==>   [by def. of eval_I(&)]
    AI, eval_I(F & -H) = 0                                    ==>   [by def. of |=]
    AI, not (I |= F & -H)                                     ==>   [by de.f of
unsatisfiable]
        F & -H is unsatisfiable.

Another answer:
        F v G |= H                                            ==>   [by def. of logical
consequence]
    AI, if I |= F v G then I |= H                             ==>   [by def. of |=]
    AI, if eval_I(F v G) = 1 then eval_I(H) = 1               ==>   [by def. of eval_I(v)]
    AI, if max(eval_I(F), eval_I(G)) = 1 then eval_I(H) = 1   ==>   [by def of max]
    AI, if eval_I(F) = 1 then eval_I(H) = 1                   ==>   [with existential
quantifier]
    not EI such that eval_I(F) = 1 and eval_I(H) = 0          ==>   [by def. of eval(-)]
    not EI such that eval_I(F) = 1 and eval_I(-H) = 1         ==>   [by def. of max]
    not EI such that max(eval_I(F), eval_I(-H)) = 1           ==>   [by def. of eval_I(&)]
    not EI such that eval_I(F & -H) = 1                       ==>   [by def. of
unsatisfiable]
        F & -H unsat

1b) Is it true that, for any two propositional formulas F and G, if F |= G
    and G is satisfiable, then F is satisfiable?  If it is, prove it.
    If it is not, give a concrete counterexample (and check it is so).

Answer:
    It is false.
    Counter example: Let F = p & -p, which is unsatisfiable, and G = p which is
satisfiable (e.g., I(p) = 1 is a model).
    Then we have also F |= G (any model of F is a model of G) since F is unsatisfiable.


--------------------------------------------------------------------------------------------------

Problem 2. (2.5 points).

2a) Write all clauses (as disjuctions of literals) obtained by applying Tseitin’s
    transformation to the formula (p & (q v -r)) v q.
```

Use auxiliary variables named e0, e1, e2, ... (where e0 is for the root).

Answer:

```
                 v (e0)
               /    \
             & (e1)   q
           /   \
          p      v (e2)
               /   \
              q     - (e3)
                    |
                    r
```

```
 Using the variable e3 we obtain 12 clauses:          and without e3:
 1) e0                                              1) e0

 2) -e0 v e1 v q    [e0 -> e1 v q]                  2) -e0 v e1 v q    [e0 -> e1 v q]
 3) -e1 v e0        [e0 <- e1]                      3) -e1 v e0        [e0 <- e1]
 4) -q v e0         [e0 <- q]                       4) -q v e0         [e0 <- q]

 5) e1 v -p v -e2   [e1 <- p & e2]                  5) e1 v -p v -e2   [e1 <- p & e2]
 6) -e1 v p         [e1 -> p]                       6) -e1 v p         [e1 -> p]
 7) -e1 v e2        [e1 -> e2]                      7) -e1 v e2        [e1 -> e2]

 8) -e2 v q v e3    [e2 -> q v e3]                  8) -e2 v q v -r    [e2 -> q v -r]
 9) e2 v -q         [e2 <- q]                       9) e2 v -q         [e2 <- q]
 10) e2 v -e3        [e2 <- e3]                     10) e2 v r         [e2 <- -r]

 11) -e3 v -r        [e3 -> -r]
 12) e3 v r          [e3 <- -r]
```

2b) Prove that it is not true that for any propositional formula F, F and Tseitin(F) are logically equivalent.

Answer:
Let F be -p. Then Tseitin(F) is the set of clauses {e0, -e0 v -p, e0 v p}.
Now we have

```
p  e0 ||  F  |  Tseitin(F)
--------------------------
0  0  ||  1  |     0  <--    I(p) = 0, I(e0) = 0 is a model of F, but not of Tseitin(F)
0  1  ||  1  |     1
1  0  ||  0  |     0
1  1  ||  0  |     0
```

2c) Is 3-SAT NP-complete? Explain your answer very briefly, using the fact that SAT (deciding the satisfiability of an arbitrary propositional formula F) is NP-complete

Answer:
3-SAT is the problem of deciding the satisfiability of a set of clauses S with at most 3 literals per clause.
Yes, it is NP-complete. For this, we need to show two things:
A) 3-SAT is NP-hard (not easier than general SAT) since we can polynomially reduce SAT to 3-SAT: the Tseitin transformation in polynomial time reduces SAT for an arbitrary F to a polynomial-sized 3-SAT set of clauses S that is equisatisfiable to F.
   Since SAT is NP-hard, so is 3-SAT.
B) 3-SAT is in NP (not harder than SAT) since one can polynomially reduce 3-SAT to

SAT:
this is trivial because 3-SAT is already a particular case of SAT.

Problem 3. (2.5 points).

3) Given S a set of clauses (CNF) over *n* propositional symbols,
and Resolution the deductive rule:

```
            p v C          -p v D
            ---------------------      for some symbol p
                    C v D
```

Let Res(S) be its closure under resolution. For each one of the following cases, indicate whether Res(S) is infinite or finite, and, if finite, express an accurate upper bound on its size in terms of n. Very briefly explain why. Use the notation a^b to express exponentiation.

3a) S is a set of Horn clauses.

Answer:
Resolution is closed under Horn clauses.
The Horn clauses can be:
a) clauses with only negative literals : 2^n
b) clauses with one positive literal, and the rest negatives : n*2^(n-1)
An upper bound on |Res(S)| is 2^n + n*2^(n-1)

3b) If clauses in S have at most two literals.

Answer:
Resolution is closed under Krom clauses (clauses with at most two literals)
The Krom clauses can be:
a) clauses with no literals (the empty clause) : binomial(2n, 0) = 1
b) clauses with one literal : binomila(2n, 1) = 2n
c) clauses with two literals : binomial(2n, 2) = 2n(2n-1)/2
An upper bound on |Res(S)| is 1 + 2n + 2n(2n-1)/2

3c) S is an arbitrary set of propositional clauses.

Answer:
Resolution is closed under propositional clauses.
There are at most 2^(2n) different propositional clauses (4^n)
This is an upper bound on |Res(S)|.

Problem 4. (2.5 points).

4a) Prove that for any propositional formula F (that is, built up with AND (&), OR (v) and NOT (-) connectives), there exists a logically equivalent formula G which is built up with exclusively NAND connectives (and propositional symbols).

We recall that the NAND connective is defined as NAND(x, y) = -(x & y).

Answer:
We only need to express the AND, OR and NOT connectives in terms of NAND connectives.

First, we note that
NOT(x) = -(AND(x, x))) = NAND(x, x).
Moreover,
AND(x, y) = -NAND(x, y) = NAND(NAND(x, y), NAND(x, y)).
Finally,
OR(x, y) = --(x v y) = -(-x & -y) = NAND(-x, -y) = NAND(NAND(x, x), NAND(y, y))

4b) Let P be a fixed set of propositional symbols.

Given two interpretations I, I': P -> {0, 1}, we write
I <= I' iff I(p) <= I'(p) for all p in P.
We say a formula F is MONOTONIC iff I <= I' implies that eval_I(F) <= eval_I'(F).

Prove that any propositional formula F built up only with AND & and OR v connectives is monotonic. Hint: use induction on F.

Answer:
Base case. If F is a propositional symbol p, then eval_I(F) = I(p) <= J(p) = eval_J(F).

Inductive case.
If F is of the form G & H, then by induction hypothesis
eval_I(G) <= eval_J(G) and eval_I(H) <= eval_J(H). Hence

eval_I(F) = min(eval_I(G), eval_I(H)) <= min(eval_J(G), eval_J(H)) = eval_J(F).

And similarly, if F is of the form G | H, then by induction hypothesis
eval_I(G) <= eval_J(G) and eval_I(H) <= eval_J(H). Therefore

eval_I(F) = max(eval_I(G), eval_I(H)) <= max(eval_J(G), eval_J(H)) = eval_J(F).