1. [20 points]
Consider a formal syntax for well-formed propositional logic given as:

\[ Formulas \alpha, \beta ::= p_i \mid \neg \alpha \mid \alpha \lor \beta \mid \alpha \land \beta \]

where \( P = \{p_0, p_1, p_2, \ldots\} \) is an infinite set of propositions, and where \( i \in \mathbb{N} \).

For any formula \( \alpha \), let the dual of \( \alpha \) be defined as follows, inductively:

\[
\begin{align*}
\text{Dual}(p) &= \neg p \text{ for every } p \in P \\
\text{Dual}(\neg \alpha) &= \text{Dual}(\alpha) \text{ for every formula } \alpha \\
\text{Dual}(\alpha \lor \beta) &= \text{Dual}(\alpha) \land \text{Dual}(\beta) \text{ for any two formulas } \alpha, \beta \\
\text{Dual}(\alpha \land \beta) &= \text{Dual}(\alpha) \lor \text{Dual}(\beta) \text{ for any two formulas } \alpha, \beta
\end{align*}
\]

Prove that for any model \( M \) and any formula \( \alpha \),

\[ M \models \alpha \text{ iff } M \not\models \text{Dual}(\alpha) \]

You must prove this formally by structural induction over formulas (using the inductive definition of \( \models \) presented in class). Vague arguments or hand-waving arguments will fetch close to zero points. Wrong boring proofs are fine (don’t try to be concise, try to be correct and formal first).

2. [10+10+10+10=40 points]
Consider the Dafny program provided on the website: Product1, Product2, Half, and DividesIt

Prove them correct using Dafny by writing down appropriate loop invariants. You are allowed to only add (side-effect free) annotations to the code—you cannot change the executing code in any way (not even in trivial ways that may seem correct to you).

Submit your program (with proof that Dafny verified it) using a hardcopy printout, and also email your (verified) programs to the TA in a tarball.

A few notes:

- Product1 and Product2 are inefficient ways to compute the product. Aim is to get you started thinking about inductive invariants and writing them formally in logic and in Dafny.
- Note that we are using the type \texttt{nat} (natural numbers), a subtype of
integers, rather than integers here.
Dafny will hence check that every assignment to a natural number
necessarily assigns a non-negative number.
Also, in Dafny, int and nat types are pure mathematical integers and
natural numbers, and don’t overflow, etc.

• Half is a program that uses integer division and asks you to write
invariants using them. Note the use of "/" (integer division) and
"%" (modulo) operators.

• In DividesIt, Dafny needs a lemma to prove the program correct.
   Lemmas are written as procedures. We have already provided the
   lemma (make sure you understand it and you agree it’s valid).
   We don’t prove the lemma but just assume it; later in the course, we
   will see how to prove lemmas.
   Also, this program has a decreases clause to prove the loop termi-
   nates— ignore understanding this for now. But keep it there so that
   Dafny does not complain.
   In the other programs above, Dafny actually uses a simple heuristic
to reason that the loops terminate, and hence doesn’t require a
decreases clause.

3. [15+5=20 points]
Consider the Dafny program twoxplusy.dfy (ensure it verifies in Dafny).
For this program, write down the three verification conditions (using meta-
steps).
Your verification conditions should be purely in logic (assuming natural
semantics of Dafny programs) and, of course, should be valid. Also, as-
suming that the invariant is changed so as to drop the conjunct "0 <= n",
write down the precise verification condition that becomes invalid.
(BTW, Dafny would still prove the program correct as it knows that n ≥ 0
is an invariant using some heuristics.)