INTRO. TO LOGIC & FUNCT. PROG.

In this programming assignment, you will take the data type of propositions defined in class and write simple programs to manipulate them.

type prop = **P** of string | **T** | **F**

| **Not** of prop | **And** of prop * prop

| **Or** of prop * prop | **Implies** of prop * prop

;;

The functions you need to implement are:

1. **height**: prop -> int, which returns the height of a proposition (height of the operator tree, counting from 0).
2. **size**: prop ->, which returns the number of nodes in a proposition (number of nodes in the operator tree).
3. **letters**: prop -> string set, which returns the *set* of propositional variables that appear in a proposition.
4. **truth**: prop -> (string -> bool) -> bool, which evaluates a proposition with respect to a given truth assignment to the propositional letters.
5. **nnf**: prop -> prop, which converts a proposition into negation normal form, where all **Not**'s appear just above only propositional letters, and strictly below **And**'s and **Or**'s, and all **Implies** have been replaced by logically equivalent forms.
6. **cnf**: prop -> (prop *set set)*, which converts a proposition into conjunctive normal form (POS) as a (conjunctive) *set* of *clauses*, where each clause is considered as a (disjunctive) *set* of *literals* (which are either propositional letters or their negation). Note: Literals are a subset of prop.
7. **dnf**: prop -> (prop *set set*), which converts a proposition into disjunctive normal form (SOP) as a (disjunctive) set of *terms*, where each term is a *set* of *literals (which are either propositional letters or their negation). Literals are a subset of prop.*
8. **isTautology**: prop -> bool, which checks if a proposition is a tautology.
9. **isContradiction**: prop -> bool, which checks if a proposition is a contradiction.

10. **isSatisfiable**: prop -> bool, which checks if a proposition is satisfiable.
11. **isEquivalent**: prop -> prop -> bool, which checks if two propositions are logically equivalent.
12. **entails**: prop -> prop -> bool, which checks if the second proposition is a logical consequence of the first proposition.

You may like to also have functions that convert a set of set of literals (output of cnf or dnf) into a prop (maxterm/minterm)

You will need to create enough examples and show that the **truth** of any proposition with respect to a truth assignment is preserved when converting it to a normal form. You may also like to show that if **entails** *p1 p2* then **isTautology** (**Implies** *p1 p2*), and other such natural laws.
