QUESTION 1:

(a) Syntax: developer uses compiler to translate C code into executable program.
Semantics: program can be run on a Linux computer with an Intel processor.

(b) Syntax: using artificial intelligence to generate two sentence summaries.
semantics: publish and share the results with subscribers.

(c) Syntax: use a voice activated assistant.
Semantics: turn on the lights.

(d) Syntax: submit the information for their tax returns via online system.
Semantics: find error and submit corrected version.

QUESTION 2:

(a) For a(one) digital logic circuit, it has 1 truth table.
But, totally there is only **8** possibility truth tables for two variables, A and B.

```
PropScratch> truthTable (AND (VAR "A") (VAR "B"))
  A   |   B   |
------+-------+------
False | False | False
------+-------+------
False | True  | False
------+-------+------
True  | False | False
------+-------+------
True  | True  | True

PropScratch> nub (vars (AND (VAR "A") (VAR "B")))
["A","B"]
PropScratch> picTree (AND (VAR "A") (VAR "B"))
AND
.-.
| |
A B

PropScratch> truthTable (AND (NOT(VAR "A")) (VAR "B"))
  A   |   B   |
------+-------+------
False | False | False
------+-------+------
False | True  | True
------+-------+------
True  | False | False
------+-------+------
True  | True  | False

PropScratch> nub (vars (AND (NOT(VAR "A")) (VAR "B")))
["A","B"]
PropScratch> picTree (AND (NOT(VAR "A")) (VAR "B"))
 AND
 .--.
 |  |
NOT B
 |
 A
```

```
PropScratch> truthTable (AND (NOT(VAR "A")) (NOT(VAR "B")))
  A    |   B    |
-------+--------+-------
False  | False  | True
-------+--------+-------
False  | True   | False
-------+--------+-------
True   | False  | False
-------+--------+-------
True   | True   | False

PropScratch> nub (vars (AND (NOT(VAR "A")) (NOT(VAR "B"))))
["A","B"]
PropScratch> picTree (AND (NOT(VAR "A")) (NOT(VAR "B")))
   AND
  .---.
  |   |
 NOT NOT
  |   |
  A   B

PropScratch> truthTable (AND (VAR "A") (NOT(VAR "B")))
  A    |   B    |
-------+--------+-------
False  | False  | False
-------+--------+-------
False  | True   | False
-------+--------+-------
True   | False  | True
-------+--------+-------
True   | True   | False

PropScratch> nub (vars (AND (VAR "A") (NOT(VAR "B"))))
["A","B"]
PropScratch> picTree (AND (VAR "A") (NOT(VAR "B")))
AND
.--.
|  |
A NOT
   |
   B
```

```
PropScratch> truthTable (OR (VAR "A") (VAR "B"))
  A   |   B   |
------+-------+------
False | False | False
------+-------+------
False | True  | True
------+-------+------
True  | False | True
------+-------+------
True  | True  | True

PropScratch> nub (vars (OR (VAR "A") (VAR "B")))
["A","B"]
PropScratch> picTree (OR (VAR "A") (VAR "B"))
OR
.-.
| |
A B

PropScratch> truthTable (OR (VAR "A") (NOT(VAR "B")))
  A   |   B   |
------+-------+------
False | False | True
------+-------+------
False | True  | False
------+-------+------
True  | False | True
------+-------+------
True  | True  | True

PropScratch> nub (vars (OR (VAR "A") (NOT(VAR "B"))))
["A","B"]
PropScratch> picTree (OR (VAR "A") (NOT(VAR "B")))
OR
.--.
|  |
A NOT
   |
   B
```

```
PropScratch> truthTable (OR (NOT(VAR "A")) (NOT(VAR "B")))
  A    |   B    |
-------+--------+-------
False  | False  | True
-------+--------+-------
False  | True   | True
-------+--------+-------
True   | False  | True
-------+--------+-------
True   | True   | False

PropScratch> nub (vars (OR (NOT(VAR "A")) (NOT(VAR "B"))))
["A","B"]
PropScratch> picTree (OR (NOT(VAR "A")) (NOT(VAR "B")))
   OR
  .---.
  |   |
 NOT NOT
  |   |
  A   B

PropScratch> truthTable (OR (NOT(VAR "A")) (VAR "B"))
  A    |   B    |
-------+--------+-------
False  | False  | True
-------+--------+-------
False  | True   | True
-------+--------+-------
True   | False  | False
-------+--------+-------
True   | True   | True

PropScratch> nub (vars (OR (NOT(VAR "A")) (VAR "B")))
["A","B"]
PropScratch> picTree (OR (NOT(VAR "A")) (VAR "B"))
 OR
 .--.
 |  |
NOT B
 |
 A
```

(b) These are unlimited digital logic circuits that use two variables, A and B to generate one of 8 possibility truth table.

Because it is possible for two different Prop values to have the same truth table. Plus, A and B can used multiple times which not effect the result of truth table. The reason is reduction: any digital logic circuits could be reduced to a normal form (TRUE or FALSE). Then, a normal form also could generate unlimited digital logic circuits.

QUESTION 3:

(a) Yes. By using listProps (normalize [ ] t1) for t1
which I build t1 = AND (OR FALSE TRUE) (NOT TRUE)

```
PropScratch> listProps (normalize [] t1)
1) AND (OR FALSE TRUE) (NOT TRUE)
2) AND TRUE (NOT TRUE)
3) AND TRUE FALSE
4) FALSE
```

(b) No. Because every reduction sequence terminates after at most one step for any t. (in Mark's discussion)

(c) Yes. OR expression could reduced to different normal form by do different reduction sequences. According Mark's discussion, "The expression `OR TRUE FALSE` could be reduced to either `TRUE` or `FALSE`, depending on which of the two reduction rules we apply to the initial expression.()"