Assignment 1: Runtime Verification of Properties specified in Propositional Logic

Design and develop a tool that constructs an RV monitor that verifies a given property specification, and then engages the monitor to verify a given signal trace.

Formats of Input and Output

Input

1. Property file
   Format of property file: a text file with a single line of text that contains the PL formula to be verified. May contain any of the following operators: not (!), and (^), or (v), implies (>), and brackets ('(' and ')'). A valid name for an atomic proposition is any sequence of English alphabets (capital or small).
2. Instrumented signal file
   Format of instrumented signal file: First line contains the names of the atomic propositions in comma-separated style. Line *i* (*i > 1*) contains the truth values (1 for true, 0 for false) of each of the atomic propositions (in the same format as the first line) at time *i-2* (time is counted from *0*).

Output

1. Verdict file
   Format of verdict file: line *i* contains the truth value of the verdict at time *i-1* .

Working

- Running `construct_monitor.sh test.property` must
  - read the property file `test.property`,
  - parse the formula into a tree,
  - generate a C program `test_monitor.c` .
- Running `perform_RV.sh test.property system1.input system1.verdict` must
  - run `construct_monitor.sh test.property`
  - compile `test_monitor.c` using simple `gcc` to give `test_monitor.out`

    - run `test_monitor.out` to read the instrumented signal trace from `system1.input`, perform RV of the property, and record the verdicts in `system1.verdict`

Submission

- Upload a single zip file `<roll_number>_assignment1.zip` that contains `construct_monitor.sh`, `perform_RV.sh`, and any other source files that you use. The submission will be tested on a standard linux machine that has support for bash, C, C++, Java, Python. Do not use any non-standard libraries.

Other points to consider

- The submissions made by the class will be compared with each other on the basis of runtime.
- Use git or any other version control system.
- Future assignments will build on this one. So code wisely!