**Testing**

**Parser**

The first thing to test was the Parser of the config files. We did this by initializing the ConifgParser with various test files and then printing the parser. Printing the parser would display the router-id, input-ports, outputs and timers that were extracted from the config file, if they existed. We could then compare the output with the values we expected.

Using this method, we parsed a valid config file with no timers and also one with timers, and checked the correct values were obtained. Both cases outputted the expected results. Next was to try it with numerous config files that should not work, and check that an error was raised, and the program stopped.

We tried this with files containing no router-id, or no input-ports, or no outputs. We rearranged the order of the lines, so router-id was the last line and similar such cases. We used a filename that does not exist and an empty file. We set the router-id outside the range of 1-64000, we set the input-port and output-port outside the range of 1024-64000. We tried using duplicate input-ports or the same input-port as an output-port. In all these cases, and other cases we have not mentioned, an error message explaining the problem was displayed and the program exited.

**RIP**

The next step was to test the actual functionality of the RIP protocol.

Test routing tables converge to correct values.

Test routers react correctly to a router going down

Test router coming back online converges to the correct routing table and all other routers include this new router to their tables.

Test split horizon with poisoned reverse

Test timing of update packets unaffected