CMPT310 Assignment 2 Name: Branton Li Student#: 301311707

My code follows the requirements mentioned in the assignment. User has to input the filename for the program to read the logics rules and the user then enter a query of an atom for backward chaining.

Each rule from the file will be stored in a 1D list and there is an outer list that contains all the rule lists. And the steps taken to derive to the query atom is printed in forward chaining manner.

I implemented a couple of functions in this program and will talk about it in the following:

1. checkBody:

   The checkBody function is to check the specific cases that we may encounter when performing backward chaining.

   The first if-statement is to check if the head that we are searching for is also inside the body (e.g. p and q → p). we will not perform backward chaining on a rule that matches the above pattern.

   The second if-statement is to handle the case such that it may cause the program to loop infinitely (e.g. the pair of logics: p→q and q→p). The program records all the rules that is used throughout the deriving process, it first checks if the length of both the previous step and the rule that matches our current goal are equal to 2. Then it will check the head of the previous step with the body of the current step and the body of the previous step to the head of the rule that is being evaluated. If all the conditions are fulfilled, then the program will not expand that rule for further chaining.

2. combine:

   The combine function is used to combine the body of the rule that is being evaluated to our goal list. It will only add the variables that is not contained in the goal list and return the combined list.

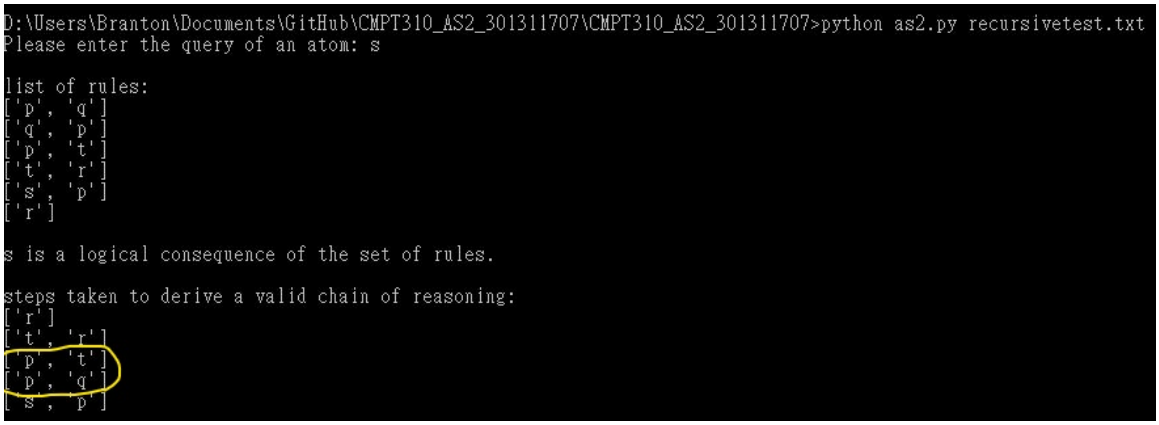
3. Solve:

   The Solve function is developed according to the given pseudo code in the pdf but with a slight twist. I have added extra checking function which is the checkBody function to handle the special cases that may corrupt the given pseudo code function.

Test Cases:

I have developed 3 test cases based on the hint of the assignment pdf.

Recursivetest.txt:



For this test case, I added the pair of logics: p→q and q→p as part of the given rules and the atom for query is s. From the above result, we can observe that the program avoids the infinite loop and produces the correct result.

As for the solve function, the program will go through the rules with a for loop and the pair of infinite loop logics will be taken as the second step. But after taking the rule p→q, the program did not expand the rule q→p and prevent itself from looping infinitely because of the checkBody function. Since q can only derived from p and there are no more rules that can be used with q as our goal, so the program just leaves the loop with q as our goal and proceed forward to check our previous goal p and produced the correct logical consequence after a couple more iterations.

Headinbodytest.txt:

For this test case, I added p and q → p as the first rule that gets inputted and the atom for query is set to p. From the above result, we can observe that the program ends without taking performing any backward chaining as expected.

With backward chaining, the program will first expand the rules with p as the header, and the only rule that matches the requirement is the rule p and q → p. But we know that this rule cannot be used / derived since our head/target is also inside the body. The program hence will not expand the rule as it is blocked by the checkBody function and returned as there are no more rules that can be expanded. Hence, the program have the correct conclusion with p is not a logical consequence of the set of given rules.

Tautologytest.txt:

For this test, a tautology q→q is added to the list of rules and our query atom should be set to q. From the above result, we can observe that the program avoids the tautology and prevent itself from looping infinitely and produced the correct output.

As stated before, the program loops through the set of rules with a for loop and the tautology will be the first rule that matches the header q. But with the checkBody function, it blocks the program to not expand on the tautology as the header that we are searching for is inside the body. Hence, the program moved on and use the other rules with a header q to continue with the backward chaining process and produce the correct output at last.