What to submit

Implement your **forward chaining** program in one file (FOL_FC.java, FOL_FC.cs, FOL_FC.py). Implement the **incremental forward chaining in another file** (FOL_IFC.java, FOL_IFC.cs, FOL_IFC.py)

Submit a .zip package with the name **"YourLastName_YourFirstName_HW3.zip"**, the package contains the following:

- (FOL_FC.jar, FOL_FC.exe or FOL_FC.py) and (FOL_ IFC.jar, FOL_ IFC.exe or FOL_ IFC.py) files for your implementation, these files should be runnable from command line. The files take only one argument which is the test case file. In case of java or C#, submit also the code (FOL_FC.java or FOL_FC.cs) and (FOL_IFC.java or FOL_IFC.cs) alongside the executable files. These executable files should print your output on the screen.

- Submit transcript files (two files for each test case one for running the FC and the other for the IFC) that contains your inferred arguments and if you found the requested argument or not. The transcript files should be named (TestCaseName_out_FC.txt) and (TestCaseName_out_IFC.txt)

- A Readme.txt file, in which mention:
    - The version of Python you used (if python used), framework used in case of C# or Java.
    - Any additional libraries needed to run your code.
    - Any additional resources, references, or web pages you've consulted.
    - List any person with whom you've discussed the assignment and describe the nature of your discussions.

- A Report.pdf or Report.doc file, in which you summarize your implementation decision and then explain how your implementation of incremental forward-chaining is more efficient.

Grading Criteria

5 (100%): The program works as expected for both implementations; has a clear README for the grader; includes a complete set of transcripts; complete report.

4 (93%): The program works as expected for both implementations, but possibly with a minor flaw; has a clear README for the grader; includes a complete set of transcripts; complete report.

3 (80%): One of the two implementations only works, OR, the program works for both implementations but one of these three requirements is not complete or missing (Clear README, Complete set of transcripts, complete report).

2 (60%): One of the two implementations only works, and one of these three requirements is not complete or missing (Clear README, Complete set of transcripts, complete report), OR, the program works for both implementations but two of these three requirements are not complete or missing (Clear README, Complete set of transcripts, complete report).

1 (40%): None of the implementations is working but a serious attempt has been made on the assignment, or the three requirements is not complete or missing (Clear README, Complete set of transcripts, complete report).