8.	¬D(Penelope) ∧ ¬M(Penelope)	Commutative law, 7
9.	¬D(Penelope)	Simplification, 8

1.14 Assignment: Truth Table Generator

Are you tired of writing out all of these truth tables by hand? Do you wish you could use what you have been training for to automate this tedious task? Well look no further. The Truth Table Generator is here!

Concept

In order to avoid the extra complication of variable names and English statements, we will restrict our domain of inputs to the symbols `p`, `q`, and `r`. We will allow the user to enter an argument consisting of one or more hypotheses and a conclusion, generate a truth table for each logical statement, and then determine whether or not the argument is valid:

```
Welcome to the truth table generator
Valid symbols are p, q, and r
Valid operators are \/, /\, ~, =>, and <=>
Whitespace is ignored
Example:  (p \/ q) => r
Enter a hypothesis or type "therefore" when done:  (p /\ q) => r
  p  q  r  (p /\ q) => r

  T  T  T             T
  T  T  F             F
  T  F  T             T
  T  F  F             T
  F  T  T             T
  F  T  F             T
  F  F  T             T
  F  F  F             T

Enter a hypothesis or type "therefore" when done: ~r
  p  q  r  (p /\ q) => r  ~r

  T  T  T             T   F
  T  T  F             F   T
  T  F  T             T   F
```

```
  T  F  F              T   T
  F  T  T              T   F
  F  T  F              T   T
  F  F  T              T   F
  F  F  F              T   T

Enter a hypothesis or type "therefore" when done: q
  p  q  r  (p /\ q) => r  ~r  q

  T  T  T              T   F  T
  T  T  F              F   T  T
  T  F  T              T   F  F
  T  F  F              T   T  F
  F  T  T              T   F  T
  F  T  F              T   T  T
  F  F  T              T   F  F
  F  F  F              T   T  F

Enter a hypothesis or type "therefore" when done: therefore
Enter the conclusion: ~p
  p  q  r  (p /\ q) => r  ~r  q  ~p

  T  T  T              T   F  T   F
  T  T  F              F   T  T   F
  T  F  T              T   F  F   F
  T  F  F              T   T  F   F
  F  T  T              T   F  T   T
  F  T  F              T   T  T   T
  F  F  T              T   F  F   T
  F  F  F              T   T  F   T

The argument IS valid
```


Evaluate

It is a fairly difficult problem to parse a logical statement and implement the correct order of operations, which is parentheses, then negation, then AND, then OR, then conditional statements. It is also outside the scope of this class. I have heavily borrowed from Stroustrup's code that I use in my CISP 400 class to write a parser for you. It uses recursive functions for each level of operator to look ahead and handle itself before returning control to a lower priority operator. If you are up for a challenge look through **parser.h**, available for download below.

As long as you `#include "parser.h"` in your code, you will have access to the function `evaluate`, which will make this entire project much more doable:

```
bool evaluate(bool p, bool q, bool r, string statement)
```

This function is completely done for you. Pass in the logical `statement` to be evaluated and the values of `p`, `q`, and `r` for the particular row that is being evaluated. It will figure out the rest and return whether the statement evaluates to T or F.

Starter Code

Starter code is given below in `main.cpp`. It contains some functions done for you, including `main`, a skeleton of the other functions that remain to be implemented, and comments throughout to detail the design.

Scoring

Each function will be unit tested and your final output will tested. Upload your source file and click the submit button to receive details about the tests being performed and their results. **You can submit as many times as you want and your highest score before the due date will be kept.**

The tester will ignore whitespace in this project, but you should try to space it nicely anyway.

Note

Please set your compiler to use C++11 or above in order to get parser.h to compile.

178542.377452.qx3zqy7

LAB ACTIVITY | 1.14.1: Assignment: Truth Table Generator | 15 / 15

Submission Instructions

Downloadable files

`main.cpp` and `parser.h` **Download**

Compile command

```
g++ main.cpp -Wall -o a.out
```

We will use this command to compile your code

Upload your files below by dragging and dropping into the area or choosing a file on your hard drive.