UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

CSE 015: Discrete Mathematics
Laboratory 3

Fall 2021

Preliminary Notes

- **This lab must be solved individually.** You can discuss your ideas with others, but when you prepare your solution you must work individually. Your submission must be yours and yours only. No exceptions, and be reminded of the CSE academic honesty policy discussed in class.
- Your solution must be exclusively submitted via CatCourses. Email submission will not be accepted. Pay attention to the posted deadline because **the system automatically stops accepting submissions when the deadline passes. Late submissions will receive a 0**. You can upload one or more `.py` files.
- Start early.

Introduction

We will use an updated version of `logic.py` that offers additional features. Make sure you download the new version from CatCourses and place it in your working directory. **The version from last week will not allow you to solve this exercise.**

Originally, to create a truth table for the expressions $p \wedge q$, and $p \vee q$, we had to do the following:

```
myTable = TruthTable(['p', 'q'], ['p and q', 'p or q'])
```

We no longer have to provide the first parameter, so now we call:

```
myTable = TruthTable(['p and q', 'p or q'])
```

The library goes through the list of propositions and figures out the variable names for us.

To solve the questions in this exercise, you have to examine the structure of the truth table built by the object `TruthTable`. This can be done by accessing the data member `table`. For example, the following lines extract the truth table and print it to the screen.

```
myTable = TruthTable(['p and q', 'p or q'])
a = myTable.table
print(a)
```

The truth table is represented by a list of lists, with each sublist representing a row in the truth table. To solve the following questions, it will be useful for you to first examine its structure and interact with it from the command line (e.g., to read out its elements).

Tautologies, contingencies and contradictions

Write a program that reads a logic expression from the keyboard and prints to the screen whether the expressions is a tautology, a contingency, or a contradiction. Your code should work for expressions with an arbitrary number of variables (i.e., do not assume the expression has only two variables and therefore the truth table has just four rows).

Optional – Logical Equivalences

This question will not give you extra credit, but you are invited to try it out if you finish early. Write a program that reads and arbitrary number of logical expressions and determines if they are all logically equivalent to each other.