Discrete Structures

Truth Table Generator

Allen Williams

Instructions for use: The user should enter a well-formed propositional logic expression using the standard keyboard keys associated with logical connectives using letters A-Z to represent propositions, as indicated by the prompt. It is important to enter a well-formed expression as the program currently does no input checking. Parentheses should be placed around sub-expressions to indicate the correct order, otherwise the program will assume an order from right to left.

Cases which may cause program failure: The user must enter a well-formed expression or the program will fail. There is a small problem with the formatting of the text in the text file because tabs after strings of different lengths will not always line up, so the truth values and the expressions to which they correspond do not line up in nice vertical columns. I tried to find a way to compile .tex files into .pdfs over the internet but was unable to do so in time to turn in this project, so if the user does not have LaTeX installed in the same location that it is on my computer, the program will generate a .tex file, but will not be able to compile it.

General Description: The program takes a string representing an expression in propositional logic from the user. It replaces convenient keyboard keys with the Unicode value of the more appropriate symbols those keys stand in for. It first places all letters a-z or A-Z in an ArrayList, with duplicates not allowed to get the propositions. It then takes the string and converts it to postfix notation, which is much easier to evaluate. The process of converting the string to postfix generates a queue of characters. This queue is copied each time it is used because the process of dequeueing elements destroys the queue. The queue is iterated over with each proposition being stored on a stack, and when an operator is reached the appropriate number of values are popped and a new string is generated and pushed back on the stack. This value is also stored in an arraylist holding intermediate steps. The next time through a copy of the postfix queue is used with each proposition or intermediate step being stored in a HashMap which maps the symbol to its truth value. These are output and also stored in a matrix which is used for file writing and simplification. The expression is simplified by taking the minterms from the truth table and using a form of the Quine-McCluskey algorithm to attempt to minimize the Boolean expression. An attempt was made to determine if the expression was already as simple as possible, but sometimes in the minimization process the order of operands is permuted so the program will not recognize that the expression is already simplified. The program also generates a .tex file which can be used to view a very nice truth table.