```
ALGORITHM Propositional Logic Calculator
FUNCTION isWellFormed(P)
   bracketLevel = 0
   FOR c IN P DO
      IF c == "(" THEN
         bracketLevel += 1
      END IF
      IF c == ")" THEN
         IF bracketLevel == 0 THEN
            RETURN False
         END IF
         bracketLevel -= 1
      END IF
   END FOR
   RETURN bracketLevel == 0
END FUNCTION

FUNCTION parseNegation(P, truthValues)
   RETURN NOT parseProposition(P, truthValues)
END FUNCTION

FUNCTION parseConjunction(P, Q, truthValues)
   RETURN parseProposition(P, truthValues) AND parseProposition(Q, truthValues)
END FUNCTION

FUNCTION parseDisjunction(P, Q, truthValues)
   RETURN parseProposition(P, truthValues) OR parseProposition(Q, truthValues)
END FUNCTION

FUNCTION parseConditional(P, Q, truthValues)
   RETURN (NOT parseProposition(P, truthValues)) OR parseProposition(Q,
truthValues)
END FUNCTION

FUNCTION parseBiconditional(P, Q, truthValues)
   RETURN parseProposition(P, truthValues) == parseProposition(Q, truthValues)
END FUNCTION

FUNCTION parseProposition(P, truthValues)
   P = P.replace(" ", "")
```

```
IF NOT isWellFormed(P) THEN
   RETURN "Error"
END IF
WHILE P[0] == "(" AND P[-1] == ")" AND isWellFormed(P[1:len(P) - 1]) DO
   P = P[1:len(P) - 1]
END WHILE
IF len(P) == 1 THEN
   RETURN truthValues[P]
END IF
bracketLevel = 0
FOR i = len(P)-1 DOWN TO 0 DO
   IF P[i] == "(" THEN
      bracketLevel += 1
   END IF
   IF P[i] == ")" THEN
      bracketLevel -= 1
   END IF
   IF P[i] == "→" AND bracketLevel == 0 THEN
      RETURN parseConditional(P[0:i], P[i + 1:], truthValues)
   END IF
   IF P[i] == "↔" AND bracketLevel == 0 THEN
      RETURN parseBiconditional(P[0:i], P[i + 1:], truthValues)
   END IF
END FOR
bracketLevel = 0
FOR i = len(P)-1 DOWN TO 0 DO
   IF P[i] == "(" THEN
      bracketLevel += 1
   END IF
   IF P[i] == ")" THEN
      bracketLevel -= 1
   END IF
   IF P[i] == "∨" AND bracketLevel == 0 THEN
      RETURN parseDisjunction(P[0:i], P[i + 1:], truthValues)
   END IF
END FOR
bracketLevel = 0
FOR i = len(P)-1 DOWN TO 0 DO
   IF P[i] == "(" THEN
      bracketLevel += 1
```

```
        END IF
        IF P[i] == ")" THEN
            bracketLevel -= 1
        END IF
        IF P[i] == "∧" AND bracketLevel == 0 THEN
            RETURN parseConjunction(P[0:i], P[i + 1:], truthValues)
        END IF
    END FOR
    bracketLevel = 0
    FOR i = len(P)-1 DOWN TO 0 DO
        IF P[i] == "(" THEN
            bracketLevel += 1
        END IF
        IF P[i] == ")" THEN
            bracketLevel -= 1
        END IF
        IF P[i] == "¬" AND bracketLevel == 0 THEN
            RETURN parseNegation(P[i + 1:], truthValues)
        END IF
    END FOR
    RETURN "Error"
END FUNCTION

FUNCTION writeTruthTable(P)
    truthValues = {}

    FOR i FROM 0 TO length of P
        IF P[i] is a letter from "A" to "Z"
            SET truthValues[P[i]] = True
        END IF
    END FOR

    SET output to a new StringIO object
    SET sys.stdout to output

    FOR EACH statement IN keys of truthValues
        PRINT statement, " | "
    END FOR
    PRINT P
```

```
FOR EACH truthValue IN values of truthValues
   IF truthValue is True
      PRINT "T", " | "
   ELSE
      PRINT "F", " | "
   END IF
END FOR

IF parseProposition(P, truthValues) is True
   PRINT "T"
ELSE
   PRINT "F"
END IF

SET j to length of values of truthValues - 1

WHILE True in values of truthValues
   SET variable to the key at index j in keys of truthValues
   SET truthValues[variable] to not truthValues[variable]

   IF truthValues[variable] is False
      FOR EACH truthValue IN values of truthValues
         IF truthValue is True
            PRINT "T", " | "
         ELSE
            PRINT "F", " | "
         END IF
      END FOR

      IF parseProposition(P, truthValues) is True
         PRINT "T"
      ELSE
         PRINT "F"
      END IF

      SET j to length of values of truthValues - 1
   ELSE
      SET j to j - 1
   END IF
END WHILE
```

```
    SET sys.stdout to the original stdout object
    RETURN the value of output as a string
END FUNCTION

FUNCTION Conjunction()
    Press the "∨" key
    Release the "∨" key
END FUNCTION

FUNCTION Conjuction()
    Press the "∨" key
    Release the "∨" key
END FUNCTION

FUNCTION Disjunction()
    Press the "∧" key
    Release the "∧" key
END FUNCTION

FUNCTION Conditional()
    Press the "→" key
    Release the "→" key
END FUNCTION

FUNCTION Biconditional()
    Press the "↔" key
    Release the "↔" key
END FUNCTION

FUNCTION Negation()
    Press the "¬" key
    Release the "¬" key
END FUNCTION

FUNCTION OpenP()
    Press the "(" key
    Release the "(" key
END FUNCTION
FUNCTION CloseP()
   Press the ")" key
   Release the ")" key
END FUNCTION

FUNCTION LetterP()
   Press the "P" key
   Release the "P" key
END FUNCTION

FUNCTION LetterQ()
   Press the "Q" key
   Release the "Q" key
END FUNCTION

FUNCTION LetterR()
   Press the "R" key
   Release the "R" key
END FUNCTION

FUNCTION LetterS()
   Press the "S" key
   Release the "S" key
END FUNCTION

FUNCTION LetterT()
   Press the "T" key
   Release the "T" key
END FUNCTION

FUNCTION LetterU()
   Press the "U" key
   Release the "U" key
END FUNCTION

FUNCTION LetterV()
   Press the "V" key
   Release the "V" key
END FUNCTION
```

```
FUNCTION Backspace()
   Press the Backspace key
   Release the Backspace key
END FUNCTION

CLASS ConsoleGUI
   FUNCTION __init__(master)
      // Initialize the Console GUI
      self.master = master
      self.frame = create a frame
      self.frame is packed to the top
      self.command_entry = create an entry widget for user input
      self.command_entry is packed to the bottom, filling the remaining space
      self.command_entry is set to execute the command when 'Enter' is pressed
      the focus is set to the command entry widget
      self.console_output = create a text widget to display output
      the widget is initially set to 'disabled'
      self.console_output is packed to the left, filling the remaining space

   FUNCTION execute_command()
      // Execute the user's command and display output
      get the command from the command_entry widget
      use the writeTruthTable() function to generate output based on the command
      set the console_output widget to 'normal' mode to enable writing to it
      add the output to the console_output widget
      set the console_output widget back to 'disabled' mode to prevent user input
      scroll the console_output widget to the end

   FUNCTION save_console_text()
      // Save console output to a file
      get the command from the command_entry widget
      delete the command from the command_entry widget
      use the writeTruthTable() function to generate output based on the command
      open a dialog box to choose a filename to save the output to
      if a filename is selected, create a new file and write the output to it

   FUNCTION clear_console()
      // Clear the console output and command entry widgets
      set the console_output widget to 'normal' mode to enable writing to it
      delete all text in the console_output widget
```

```
        set the console_output widget back to 'disabled' mode to prevent user input
        delete the command from the command_entry widget

    FUNCTION load_file(filename)
        // Load contents of file and return them
        open the file with the given filename
        read the contents of the file and return them

    FUNCTION load_file_handler()
        // Open a dialog box to choose a file to load and display contents in console output
widget
        open a dialog box to choose a file to load
        if a file is selected, load its contents and display them in the console_output widget

    FUNCTION run()
        // Start the main loop for the program
        start the main loop for the program

Class PropologicalGUI:
    Function __init__(self, master):
        Set self.master to master
        Set the title of self.master to "Propological"
        Set the minimum and maximum size of self.master to 506 x 600 pixels
        Create an instance of the ConsoleGUI class and assign it to self.console

        Create 15 buttons with the following parameters:
            Text: "∨", "∧", "→", "↔", "¬", "(", ")", "P", "Q", "R", "S", "T", "U", "V", "⌫"
            Font: "Cambria" size 20
            Width: 6
            Foreground (text) color: white
            Background color: maroon
            Commands: Conjunction, Disjunction, Conditional, Biconditional, Negation,
OpenP, CloseP, LetterP, LetterQ, LetterR, LetterS, LetterT, LetterU, LetterV, Backspace
        Place the buttons at the following (x, y) coordinates:
            (20, 440), (140, 440), (260, 440), (380, 440), (20, 520), (140, 520), (260, 520),
(20, 280), (140, 280), (260, 280), (380, 280), (20, 360), (140, 360), (260, 360), (380,
360)

        Create a "Done" button with the following parameters:
            Text: "Done"
            Font: "Cambria" size 20
            Width: 6
            Foreground (text) color: white
            Background color: maroon
            Command: execute_command method of self.console
        Place the button at (380, 520)

        Create a "Clear" button with the following parameters:
            Text: "Clear"
            Font: "Cambria" size 20
            Width: 6
            Foreground (text) color: white
            Background color: maroon
            Command: clear_console method of self.console
        Place the button at (380, 130)

        Create a "Save" button with the following parameters:
            Text: "Save"
            Font: "Cambria" size 20
            Width: 6
            Foreground (text) color: white
            Background color: maroon
            Command: save_console_text method of self.console
        Place the button at (380, 70)

        Create a "Load" button with the following parameters:
            Text: "Load"
            Font: "Cambria" size 20
            Width: 6
            Foreground (text) color: white
            Background color: maroon
            Command: load_file_handler method of self.console
        Place the button at (380, 10)

if __name__ == '__main__':
    Create a Tkinter root window and assign it to root
    Create an instance of the PropologicalGUI class and pass root as an argument,
assign it to app
    Call the run method of self.console
END Propositional Logic Calculator
```