Laboratory Activity 6 - GUI Design: Layout and Styling	
San Juan, Edson Ray E.	11/03/2024
CPE21S4	Prof. Maria Rizette Sayo

6. Supplementary Activity:

Task

Make a calculator program that can compute perform the Arithmetic operations as well as exponential operation, sin, cosine math functions as well clearing using the C button and/or clear from a menu bar. The calculator must be able to store and retrieve the operations and results in a text file. A file menu should be available and have the option Exit which should also be triggered when ctrl+Q is pressed on the keyboard. You may refer to your calculator program in the Desktop

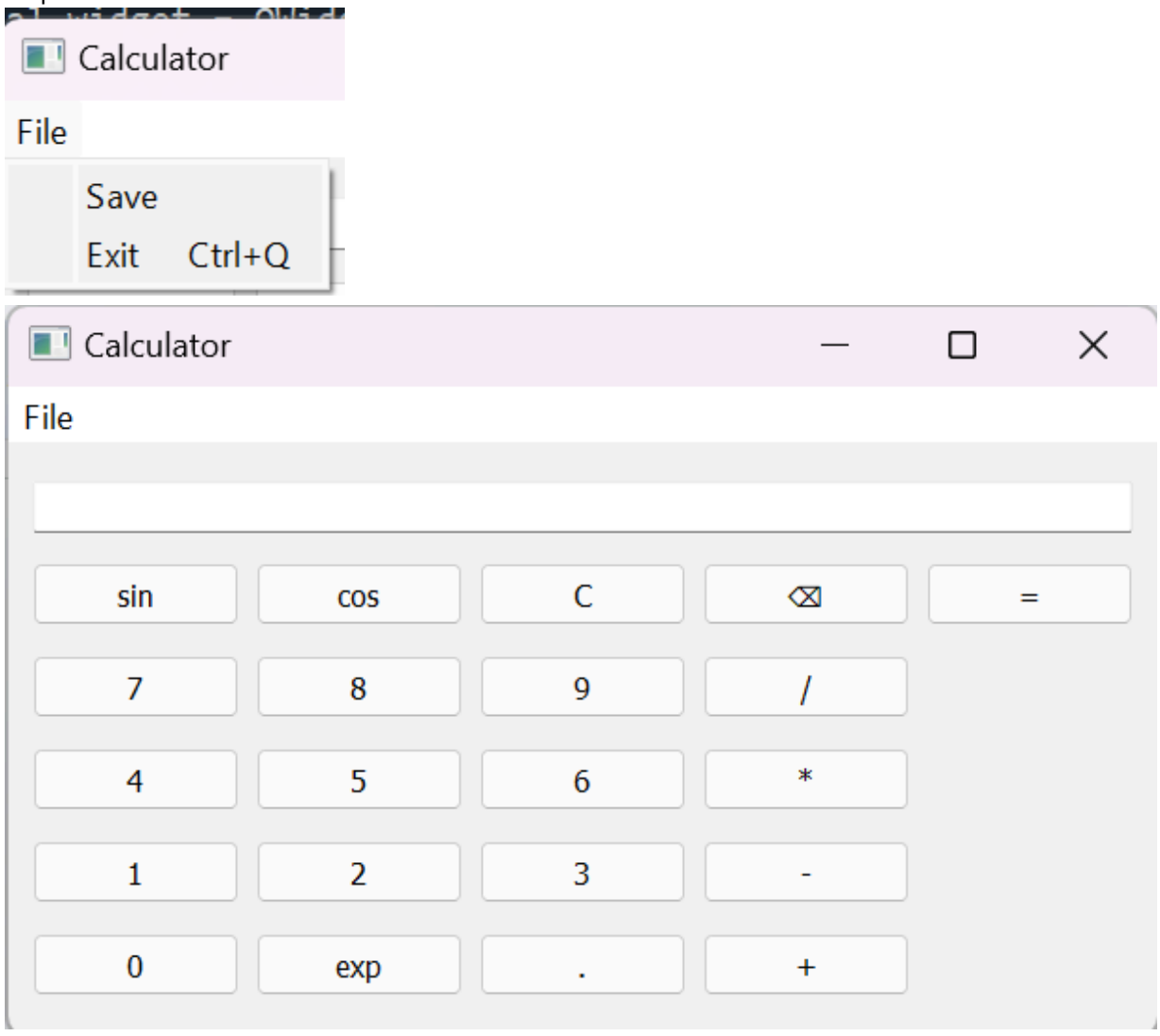
```
C/C++
import sys
import math
import re
from PyQt5.QtWidgets import (
    QApplication, QGridLayout, QLineEdit, QPushButton, QWidget,
    QMainWindow, QAction, QFileDialog, QMessageBox)
from PyQt5.QtGui import QKeySequence


class Calculator(QMainWindow):
    def __init__(self):
        super().__init__()
        self.initUI()
        self.clear_log_file()  # Clear log file on startup

    def initUI(self):
        # Create a central widget and set layout
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

        grid = QGridLayout()

        # Text display
        self.textLine = QLineEdit()
        grid.addWidget(self.textLine, 0, 0, 1, 5)

        # Button names arranged in a 5x5 grid
        names = [
            'sin', 'cos', 'C', '⌫', '=',
            '7', '8', '9', '/', '',
            '4', '5', '6', '*', '',
```

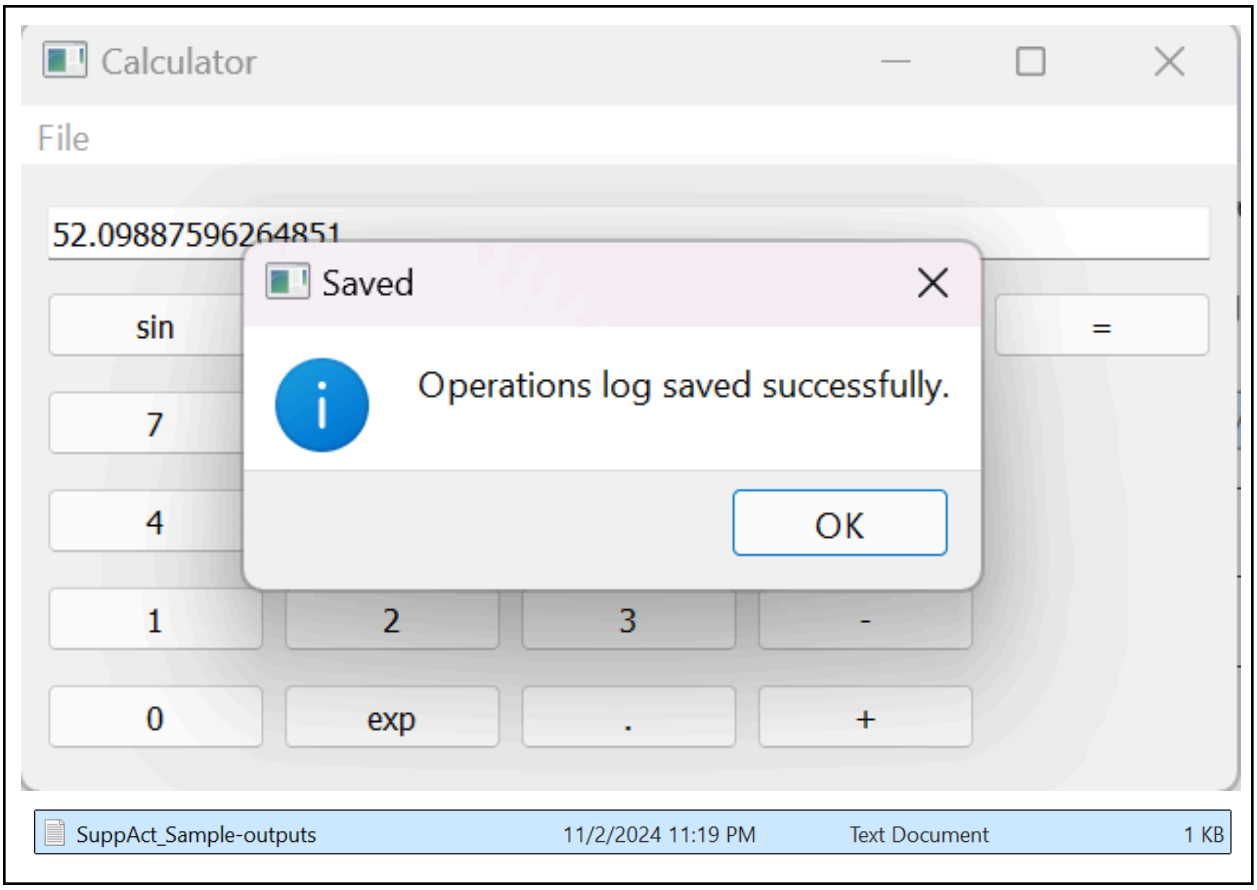
```
            '1', '2', '3', '-', '',
            '0', 'exp', '.', '+', ''
        ]

        # Add buttons to grid with updated row range to include the last row
        positions = [(i, j) for i in range(1, 6) for j in range(5)]
        for position, name in zip(positions, names):
            if name:  # Only create buttons for non-empty names
                button = QPushButton(name)
                button.clicked.connect(self.onButtonClick)
                grid.addWidget(button, *position)

        self.central_widget.setLayout(grid)

        # Set up menu bar with Save, Load, Clear, and Exit options
        menubar = self.menuBar()
        fileMenu = menubar.addMenu('File')

        saveAction = QAction('Save', self)
        saveAction.triggered.connect(self.saveToFile)
        fileMenu.addAction(saveAction)

        exitAction = QAction('Exit', self)
        exitAction.setShortcut(QKeySequence("Ctrl+Q"))
        exitAction.triggered.connect(self.exitApplication)
        fileMenu.addAction(exitAction)

        self.setGeometry(300, 300, 250, 350)
        self.setWindowTitle('Calculator')

    def onButtonClick(self):
        sender = self.sender().text()
        current_text = self.textLine.text()

        if sender == 'C':
            self.clearDisplay()
        elif sender == '⌫':
            self.textLine.setText(current_text[:-1])  # Remove the last
character
        elif sender == '=':
            original_expression = self.textLine.text()  # Save original
expression for logging
            expression = original_expression  # Copy to modify for evaluation

            # Replace 'cos <angle>' with 'math.cos(math.radians(<angle>))'
            expression = re.sub(r'(\d*)\s*\*\s*cos\s+(\d+\.?\d*)',
r'\1*math.cos(math.radians(\2))', expression)
            expression = re.sub(r'cos\s+(\d+\.?\d*)',
r'math.cos(math.radians(\1))', expression)
```

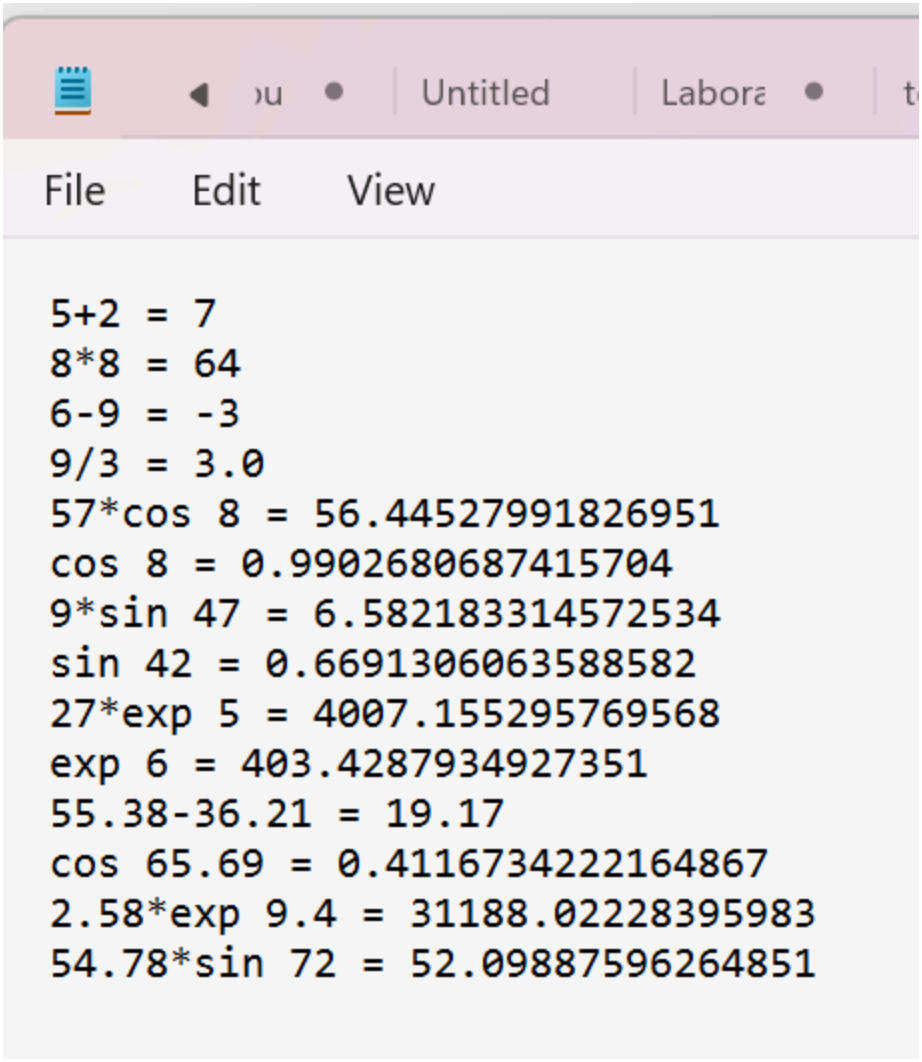
```
            # Replace 'sin <angle>' with 'math.sin(math.radians(<angle>))'
            expression = re.sub(r'(\d*)\s*\*\s*sin\s+(\d+\.?\d*)',
r'\1*math.sin(math.radians(\2))', expression)
            expression = re.sub(r'sin\s+(\d+\.?\d*)',
r'math.sin(math.radians(\1))', expression)

            # Replace 'exp <number>' with 'math.exp(<number>)'
            expression = re.sub(r'exp\s+(\d+\.?\d*)', r'math.exp(\1)',
expression)

            try:
                # Print the entire expression before evaluation
                print(f"Input: {original_expression}")

                # Evaluate the expression with access to math functions
                result = eval(expression, {"__builtins__": None}, {"math":
math})
                self.textLine.setText(str(result))
                print(f"Result: {result}")  # Print result to console
                self.logOperation(original_expression, result)  # Log the
original expression and result
            except Exception as e:
                print(e)  # Print the error for debugging
                self.textLine.setText("Error in expression")  # Handle eval
errors

        elif sender == 'sin':
            if current_text and current_text[-1].isdigit():
                self.textLine.setText(current_text + '*sin ')
            else:
                self.textLine.setText(current_text + 'sin ')
        elif sender == 'cos':
            if current_text and current_text[-1].isdigit():
                self.textLine.setText(current_text + '*cos ')
            else:
                self.textLine.setText(current_text + 'cos ')
        elif sender == 'exp':
            if current_text and current_text[-1].isdigit():
                self.textLine.setText(current_text + '*exp ')
            else:
                self.textLine.setText(current_text + 'exp ')
        else:
            self.textLine.setText(current_text + sender)


    def logOperation(self, original_expression, result):
        """Log the original expression and result to a file."""
        with open("operations_log.txt", "a") as file:
            file.write(f"{original_expression} = {result}\n")
```

```
    def clear_log_file(self):
        with open("operations_log.txt", "w") as file:
            file.write("")  # Clear the file

    def clearDisplay(self):
        """Clear the display."""
        self.textLine.clear()
        print("Display cleared")  # Indicate clearing action in the console

    def exitApplication(self):
        self.clear_log_file()  # Clear log file on exit
        self.close()  # Close the application

    def saveToFile(self):
        """Save operations log to a selected file."""
        try:
            with open("operations_log.txt", "r") as file:
                content = file.read()
            save_path, _ = QFileDialog.getSaveFileName(self, "Save Log File",
"", "Text Files (*.txt);;All Files (*)")
            if save_path:
                with open(save_path, "w") as save_file:
                    save_file.write(content)
                QMessageBox.information(self, "Saved", "Operations log saved
successfully.")
        except FileNotFoundError:
            QMessageBox.warning(self, "Warning", "No operations log to save.")

    def loadFromFile(self):
        """Load and display previous operations from log file."""
        try:
            with open("operations_log.txt", "r") as file:
                content = file.read()
            QMessageBox.information(self, "Operations Log", content)
        except FileNotFoundError:
            QMessageBox.warning(self, "Warning", "No operations log found.")

if __name__ == '__main__':
    app = QApplication(sys.argv)
    calc = Calculator()
    calc.show()
    sys.exit(app.exec_())
```

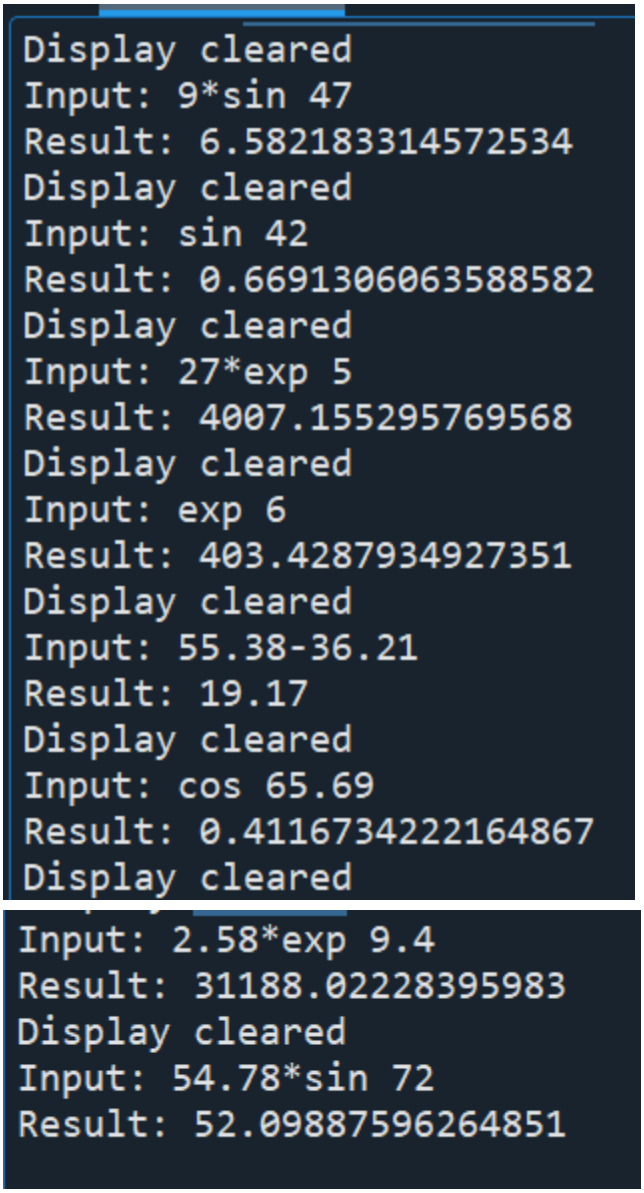
Output:


Calculator
File
Save
Exit Ctrl+Q
Calculator
File
sin
cos
C
⌫
=
7
8
9
/
4
5
6
*
1
2
3
-
0
exp
.
+


Calculator
File
52.09887596264851
Saved
Operations log saved successfully.
OK
sin
=
7
4
1
2
3
-
0
exp
.
+
SuppAct_Sample-outputs
11/2/2024 11:19 PM
Text Document
1 KB


Untitled
File
Edit
View
5+2 = 7
8*8 = 64
6-9 = -3
9/3 = 3.0
57*cos 8 = 56.44527991826951
cos 8 = 0.9902680687415704
9*sin 47 = 6.582183314572534
sin 42 = 0.6691306063588582
27*exp 5 = 4007.155295769568
exp 6 = 403.4287934927351
55.38-36.21 = 19.17
cos 65.69 = 0.4116734222164867
2.58*exp 9.4 = 31188.02228395983
54.78*sin 72 = 52.09887596264851

```
In [47]: %runfile C:/Users/eesj
SuppAct.py --wdir
Input: 5+2
Result: 7
Display cleared
Input: 8*8
Result: 64
Display cleared
Input: 6-9
Result: -3
Display cleared
Input: 9/3
Result: 3.0
Display cleared
Input: 57*cos 8
Result: 56.44527991826951
Display cleared
Input: cos 8
Result: 0.9902680687415704
```



7. Conclusion:

In conclusion, the development of this GUI-based calculator successfully integrated essential arithmetic operations, sin and cos functions, and exponential operation. The addition of functionality for clearing inputs, and saving results demonstrates a robust application that meets both practical and usability standards. The program's design allows users to perform calculations efficiently, while the logging feature provides a reliable way to track and save past computations.