Correction: PyQt5 Problem-Solving Quiz

Question 1: Debugging a Multithreaded Application (4 Marks)

Answer:

- **Potential Causes**: - The thread is not properly stopped, causing it to access invalid memory. - The main thread tries to interact with the thread after it has been deleted.

- **Solution**: - Use ‘QThread.quit()‘ and ‘QThread.wait()‘ to safely stop the thread. - Ensure the thread is properly managed using signals and slots.

Example:

```
class WorkerThread(QThread):
    def run(self):
        while not self.isInterruptionRequested():
            print("Working...")
            self.msleep(500)

# Stopping the thread safely
worker = WorkerThread()
worker.start()
worker.requestInterruption()  # Request interruption
worker.quit()
worker.wait()
```

—

Question 2: Optimizing UI Performance (4 Marks)

Answer:

- **Optimization Techniques**: - Use ‘QAbstractItemModel‘ instead of ‘QTableWidget‘ for large datasets. - Enable lazy loading (load data on demand). - Use ‘QTimer‘ to batch UI updates.

Example:

```
from PyQt5.QtCore import QTimer

# Batch updates using QTimer
timer = QTimer()
timer.timeout.connect(self.update_table)
timer.start(100)  # Update every 100ms
```

—

Question 3: Custom Signal Implementation (4 Marks)

Answer:

- **Code Example**:

```
from PyQt5.QtCore import pyqtSignal, QObject

class SignalEmitter(QObject):
    int_list_signal = pyqtSignal(list)

    def emit_signal(self):
        self.int_list_signal.emit([1, 2, 3, 4])

class SignalReceiver(QObject):
    @pyqtSlot(list)
    def receive_signal(self, int_list):
        print(f"Received: {int_list}")

# Usage
emitter = SignalEmitter()
receiver = SignalReceiver()
emitter.int_list_signal.connect(receiver.receive_signal)
emitter.emit_signal()
```

Question 4: Dynamic Layout Management (4 Marks)

Answer:

- **Code Example**:

```
from PyQt5.QtWidgets import QPushButton, QVBoxLayout, QWidget, QApplication

class DynamicLayout(QWidget):
    def __init__(self):
        super().__init__()
        self.layout = QVBoxLayout()
        self.setLayout(self.layout)
        self.add_button = QPushButton("Add Widget")
        self.add_button.clicked.connect(self.add_widget)
        self.layout.addWidget(self.add_button)

    def add_widget(self):
        widget = QPushButton("New Widget")
        self.layout.addWidget(widget)

# Usage
app = QApplication([])
window = DynamicLayout()
window.show()
app.exec_()
```

Question 5: Advanced Event Filtering (4 Marks)

Answer:

- **Code Example**:

```
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton

class EventFilter(QWidget):
```

```
    def __init__(self):
        super().__init__()
        self.button = QPushButton("Click Me", self)
        self.button.installEventFilter(self)

    def eventFilter(self, obj, event):
        if obj == self.button and event.type() == event.MouseButtonPress:
            print("Button clicked!")
        return super().eventFilter(obj, event)

# Usage
app = QApplication([])
window = EventFilter()
window.show()
app.exec_()
```