```python
import copy
import numpy as np

def implicit_qr(A):
    m, n = np.shape(A)
    W = np.zeros((m, n), dtype=complex)
    R = A.copy().astype(complex)

    for k in range(n):
        x = R[k:m, k][:, np.newaxis]
        # Determine the complex sign
        complex_sign_x1 = x[0,0]/np.abs(x[0,0]) if np.abs(x[0,0]) != 0 else 1

        e1 = np.zeros((m-k, 1), dtype=complex)
        e1[0] = 1
        v_k = x + complex_sign_x1 * np.linalg.norm(x) * e1
        v_k = v_k / np.linalg.norm(v_k)

        W[k:m, k] = v_k[:, 0]
        # Apply transformation to R
        R[k:m, k:n] = R[k:m, k:n] - 2 * np.dot(v_k, np.dot(v_k.conj().T, R[k:m, k:n]))
    return W, R
```

```python
def form_q(W):
    m, n = np.shape(W)
    Q = np.eye(m, dtype=complex)

    for k in range(n-1, -1, -1):
        v_k = W[k:m, k][:, np.newaxis]
        Q[k:m, k:m] -= 2 * np.dot(v_k, np.dot(v_k.conj().T, Q[k:m, k:m]))

    return Q


if __name__ == "__main__":
# Test

 A = np.array([[2 + 1j, 4 - 2j, 1 + 0j],
               [1 - 1j, 3 + 3j, 2 - 2j],
               [5 + 0j, 1 - 1j, 3 + 3j],
               [1 + 1j, 2 + 2j, 1 - 1j]], dtype=complex)

 W, R = implicit_qr(A)
 Q = form_q(W)

 print("Q:\n", np.round(Q))
 print("R:\n", np.round(R))
 print("Q @ R (should be close to original A):\n", np.round(Q @ R))
 print("Difference btw original A and QR:\n", np.round(np.abs(Q @ R - A)))
```