The reference paper inspires this code and aims to compare two strings represented by quantum registers. The registers a_reg[0:n] and b_reg[n, 2*n] hold two strings, where n is the bit length obtained from the DtoB function for both k and the list.
Function Description:

Quantum Bit String Comparator (QBSC):

This function compares two given strings num and k, using quantum gates. It takes four parameters:

num: The number to be compared with k.
k: The reference value against which num is compared.
num_bits: The number of bits required to represent num and k.
flist: A list to store elements that are less than k.
Quantum Circuit Initialization:

Quantum and Classical Registers:

A quantum register q is initialized, containing 2*(n + 1) qubits, where n is the number of bits.
A classical register c_reg containing two classical bits is initialized.

Ancilla Qubits (r0, r1):
Two ancilla qubits r0 and r1, are defined from the quantum register to facilitate comparison.

Binary Representation:
The binary representation of num and k is obtained using the DtoB function.

Applying X Gates:
For each bit position i in the binary representations:
If the i-th bit of k is 1, an X gate is applied to the i-th qubit in the first half of the quantum register.(a_reg)
If the i-th bit of num is 1, an X gate is applied to the i-th qubit in the second half of the quantum register.(b_reg)

Quantum Circuit Execution:
Loop for Comparison:
For each bit position i from 0 to n-1:
The apply function compares the i-th bits of num and k.
Measurements are performed on r0 and r1 to obtain comparison outcomes.
Comparison Results:

The comparison results c1, c2, c3, and c4 are obtained based on the measurement outcomes.
If c1 is not None, it indicates an inconclusive comparison result due to equality.
If c2, c3, or c4 is not None, a definite comparison result is obtained, and the loop is terminated.

Updating Final Result:

If the loop completes without yielding a definite result (flag == n), it implies that all comparison results were inconclusive, indicating equality between num and k.

Apply Function:
Function Description:

The apply function is designed to perform a comparison operation between two qubits representing individual bits of binary numbers. It applies a series of quantum gates to these qubits to determine their relationship.

Construction of the Apply Function:

Input Parameters:

qc: The quantum circuit to which gates are applied.
a: First control qubit representing the bit of the first binary number.
b: The second control qubit represents the bit of the second binary number.
r0: The first ancilla qubit used as target qubit for comparison.
r1: The second ancilla qubit used as traget qubit for comparison.
Operation Sequence:

Step 1: X Gate on Target Qubit (b):
An X gate is applied to the target qubit b. This flips the state of b.

Step 2: Controlled-Controlled-X (Toffoli) Gate:
The Toffoli gate (also known as the controlled-controlled-X gate or CCX gate) is applied to the 1st control qubit a, the 2nd control qubit b, and the target qubit r0.
This gate performs a controlled-controlled-NOT operation, flipping the state of r0 only if both a and b are in state 1.

Step 3: X Gates on a and b Qubits (a and b):
X gates are applied to both the control qubits a and b. This undoes the previous X gate operation, restoring their original state.

Step 4: Controlled-Controlled-X (Toffoli) Gate:
Another Toffoli gate is applied to a, b, and the second ancilla qubit r1.
This gate performs a similar controlled-controlled-NOT operation, flipping the state of r1 based on the states of a and b.

Step 5: X Gate on Control Qubit (a):
An X gate is applied to the control qubit a. This undoes the previous X gate operation on a, restoring its original state.

Outcome:

Applying these gates results in the appropriate state changes in the ancilla qubits r0 and r1, encoding the comparison outcome between the corresponding bits of the binary numbers represented by a and b.

Outcomes r0 and r1 are stored as C(i) = ‘00’,’01’,’10’, and ’11’.

‘00’ = both a(i) and b(i) are equal.

‘01’ = a(i) is greater than b(i).

‘10’ = b(i) is greater than a(i).

‘11’ = b(i) is greater than a(i).