Week 13 Lab

1. Sal Penza and Sam Mazarei
2. Our quantum code multiplies two binary numbers. Our code utilizes multiple AND and XOR logic gates in order to complete the multiplication process within the boundaries of quantum computing. The program uses the first 5 qubits in this particular project to represent the binary values that are being multiplied. The first two qubits represent the first binary number which in this case is 1. The next three qubits represent the second binary number which is 5. Since all qubits default with a value of zero, NOT gates are used to change certain zeros to ones to accurately represent the binary numbers we are seeking. We now use AND gates to compare the qubit values where 1 and 1, 1 and 0, or 0 and 0, outputs the same value as if they were being multiplied.

To carry out the process of multiplication of 101 x 01, we first multiply 101 by the first bit in the other binary value which would be 1. This is done by AND gates and comparing each bit of 101 with the first bit of 01. Once that is done, the second bit of 01 will be compared with every bit in 101 using AND gates. In binary multiplication, once the numbers have been multiplied, the two outputs must be added together using binary addition. This can be replicated in quantum code by using XOR gates. Since we know that the first and last bits that are multiplied are not added in binary multiplication we send that straight to output without being used in XOR logic gates. The values in between are added together using XOR gates and sent to output.

Our code was written this way because it is both simple and very easy to follow and understand. The code is very straight forward which makes it very readable.

1. The output from the simulator is static, so since 0101 in binary is 5 digitally, the output of the program is successful. The output from the quantum computer in Melbourne shows the many different probabilities of outputting anything from 0000 to 1111. In this run, it uses every combination of zeros and ones and tallies the percentage chance that each value has of being outputted. In our case, the expected output had only a 6.934% chance of being the result. Since quantum computing is still in its early stages of development, this could explain why the correct answer has such a low probability along with potential lag or other systematic limitations.