Question 1: In your own words briefly explain why the binary representation of each integer looks as it does, including how the pattern of 0's and 1's obtained changes as you go from one of these numbers to the next.

To find the binary representation of a number, you take each value of the number and divide it by 2 to find the remainder value.

So for example, for 384, you would divide by 2 and get 192 remainder 0. 192/2 has a remainder of 0, 96/2 has a remainder of 0, 48/2 has a remainder 0, 24/2 has a remainder of 0, 12/2 has a remainder of 0, 6/2 has a remainder of 0, 3/2 has a remainder of 1, and ½ has a remainder of 1.

Once the quotient of divided values is equal to 0, then you get the binary value based on the remainder values, which in this case would be 110000000, read backwards.

Then to get the complete binary value, you have to pad the value with 0s to the number of bits that is required. So if the value that you get has 9 values, you need to add 7 zeroes for a 16-bit integer. For 16-bit integers, it would be 0000000110000000, and 32-bit it would be 00000000000000000000000110000000.

Try to use the "I" option to set the integer to a 5-digit positive integer (e.g., 10000), a 10-digit positive integer (e.g., 2000000000), a 15-digit positive integer, and a 20-digit positive integer.

Question 2: Briefly explain what happens in each case.

If the number is not able to be represented in binary because there are too many remainder digits, then the decimal value for that binary value will be represented as a different value, in these cases as a negative value. For the 10000 decimal, there are enough bits to represent the value in binary. For the 10-digit value, there are not enough bits to represent the number with 16 bits, therefore when the final value is padded, the decimal value equal to the binary that is given is a different number (-27648). For the 15-digit, there are enough bits, so it can be represented as a 32-bit integer, but not a 16-bit integer, however because of padding, the value of the binary is not equal to the decimal value that is inputted. For the 20-bit, you get -1 as decimal and 1111s for binary because for integer data types, the maximum values only come to 19-digit decimals.

Question 3: What results do you get with the 16-bit integer form and with the 32-bit integer? Explain why you get each result.

Because the values are negative, the binary values are padded with 1s instead of 0s.

For -32677, the 16-bit 1000000000000010 and 32-bit 11111111111111111000000000000010
For -32767, the 16-bit 1000000000000001 and 32-bit 11111111111111111000000000000001
For -32768, the 16-bit 1000000000000000 and 32-bit 11111111111111111000000000000000

Because -32769 is out of range for short data types, the value returns to the max of that data types range, which is 32767, 16-bit as 0111111111111111, and 32-bit as 11111111111111110111111111111111. This would make the next value 32766 for short data type, which is 0111111111111110, and -32770 as the decimal for the 32-bit value which is represented by. 11111111111111110111111111111110