Problem 1:

This program sorts a set of real numbers (floats) using a radix sort algorithm. The user first enters how many numbers they'll be sorting, up to a maximum of 100, and then inputs each of the numbers. The program separates the entered numbers into two groups—positive and negative—to handle them properly in sorting. Negative numbers are processed by flipping all their bits using a bitwise operation so they can be sorted correctly using radix sort, which normally only sorts unsigned numbers. Both positive and negative groups are individually sorted by processing each bit of the numbers one at a time. After sorting, the negative numbers are converted back to their original representation by reversing the earlier bitwise operation, and finally, both groups are combined into one sorted array. The program then outputs the numbers neatly, printing each sorted number on its own line.

Problem 2:

This program sorts a list of integers (both positive and negative) using a special type of radix sort called hexadecimal radix sort. The user first provides the number of integers they want sorted, again limited to 100 at most, followed by entering each integer. To correctly handle negative integers, the program transforms each number by flipping its sign bit before sorting. The radix sort then sorts these integers by looking at them 4 bits at a time, which corresponds to one hexadecimal digit per pass. This approach requires eight passes to fully sort the 32-bit integers. After the sorting is finished, the integers are transformed back into their original form by flipping the sign bit again. Lastly, the program prints out all the sorted integers, each one on a separate line, giving the user a clearly sorted list.