CSCI 210 Assn. 2: Floating-point representation

This assignment is due at 4pm, Friday, January 30. Note that we do not have class that day: If you haven't completed your solution by the preceding class, you should drop your answers by my office. The assignment is worth 20 points.

Submit your solutions on paper, handwritten or typed. Be neat: Particularly, don't submit a sheet torn from a spiral notebook.

Robin is developing a scientific program that involves finding the sum of many `float`s together.

```
total = 0.0;
for(i = 0; i < arr_len; i++) total += arr[i];
printf("%f\n", total);
```

In the programming process, Robin discovers that the program gives very different answers based on the order of the numbers. For example (as Robin tried while examining the phenomenon), suppose an array contains one million `float`s, each the cube of a number chosen randomly between 0.0 and 1.0. For one particular example array chosen this way, Robin found the following.

random order	sum is 249962
increasing order	sum is 250096, 134 more than random order
decreasing order	sum is 249865, 97 less than random order

That is, the sum of the randomly ordered numbers was approximately 249962. But, if the program sorted them into increasing order first, the sum of the *same numbers* was 134 more!

After trying this on many arrays, Robin verified that this is not a random coincidence — consistently, the increasing sum was more than the random sum, which was more than the decreasing sum.

Explain this phenomenon. Which order of adding is most accurate, and what makes the others less accurate? (This is an essay question. I will grade your response based on correctness, clearness, and thoroughness. A particularly good answer would give some illustrative examples. I will not grade based on length; a concise, complete answer can fit into half a page.)

Hint: Consider how you would represent $1 + 2^{-100}$ in the IEEE 32-bit floating-point representation.