1. What data type in the C programming language allows for the largest values of factorial to be computed?

   If we consider about the result type to integer, we find the unsigned long long int type has 8 bytes, which can help us compute a bigger factorial range from 0 to 18,446,744,073,709,551,615, which mean it can support the computing number n from the previous 12 to 20. When n=21, it will come to value overflow.

   If we are not considering the result type as integer, we can use the long double type which give us more larger values of factorial to be computed.

2. At what input value for the recursive factorial function does your computer start to 'crash' or really slow down when you try to compute a factorial? Is it the same value as the iterative function? Experiment and report your result.

   When we choose unsigned long long int type, when n = 21, both function come to value overflow.

   When we choose long double type, when n = 26, both functions come to value overflow.

   While we are using long double type and input the number n = 1000000, the sever start to get segmentation fault on the recursive factorial function but the iterative function is still working but return a inf. This stack overflow is long after the value overflow.

   When the input number n = 100000000, the iterative function becomes a little bit slower, but the result is meaningless.

3. In 2-3 sentences, describe why you believe you saw or didn't see differences between the iterative and recursive versions of factorial.

   Because the two functions both use the multiply operation, they both arrive the top edge of the data range first before they become stack overflow at a very quick speed, so we can’t see any difference before either of them slow down.