CNS 3320 – Numerical Software Engineering

Homework 1 – Floating-point Number Systems

There are 13 questions in the homework assignment with 10 points for each question for a total of 130 points.

The following 5 questions pertain to the *normalized* floating-point number system F with B = 4, p = 2, m = -1, and M = 2:

1. How many positive numbers are there in F?
   48
2. List all of these positive numbers, in some reasonable order (in base 4, of course).
   See table below. All numbers are base 4.

mantissa	4^{-1}	4^{0}	4^{1}	4^{2}
1.0	.1	1	10	100
1.1	.11	1.1		
1.2	.12			
1.3				
2.0	.2			
2.1				
2.2				
2.3				
3.0				
3.1				
3.2				
3.3				

3. What is the smallest positive number in decimal?
   ¼
4. What is the largest number in decimal?
   60
5. What are the spacings between numbers, in decimal, in each interval bounded by the various powers of 4?
   1/16, ¼,1, 4

The next 7 questions pertain to a general, normalized floating-point system G(B,p,m,M). Express answers in terms of B, p, m, and M, as needed. Show your work where needed (you can assume as given anything from the slides).

6. What is the smallest positive number in G?
   $\boldsymbol{B^{m}}$
7. What is the largest positive number in G?
   $\boldsymbol{B^{M+1}(1-B^{-p})}$.

8. What is the spacing between the numbers of G in the range $[B^e, B^{e+1})$?
   B^{1-p+e}.

9. How many numbers of G are there in the interval $[B^e, B^{e+1})$?
   $(B-1)B^{p-1}$.

10. How many positive numbers are there in G?
    $(B-1)B^{p-1}*(M-m+1)$.
11. What is the largest positive *integer* in G?
    If $M >= p-1$ ***$B^{M+1}(1-B^{-p})$***.
    Otherwise, ***$B^{M+1} - 1$*** *(telescoping as in the exercise 7).*
12. What is the smallest positive *integer* not representable by G?
    If $M \geq p$, the smallest integer not in the system is therefore ***$B^P + 1$***. *If $M<p$, the smallest integer not in the system is* ***B^{M+1}***.

The next question has to do with a *bisection process*, which is very common in mathematical software. For example, the bisection method for finding a root of a function starts with an interval, [a, b], where f(a) and f(b) have different signs. It then computes the midpoint of the interval, $c = (a + b) / 2$. It then replaces either a or b by c so that the signs of the new f(a) and f(b) are still different, thus guaranteeing that the new interval [a, b], which is either the left or right half of the previous interval, still brackets a root. In the next 2 questions, assume B = 2 and p = 24.

13. If $a = 1$ and $b = 2$, how many times can bisection occur before there are no floating-point numbers in the interval (a, b) (in other words, a and b are adjacent floating-point numbers)?
    23 *bisections.*