1. (This question counts double) Suppose you modify Algorithm 1.6 (Binary Search) so that you set \( m \) (the “midpoint”) to \( \lfloor xb + (1 - x)t \rfloor \). (In algorithm 1.6, \( x \) is 1/2.) For the following subquestions, you may ignore the floor functions; assume the table size is such that the floor function has no effect.
   a. What is the worst-case time of the modified algorithm?
   b. What is the best-case time of the modified algorithm?
   c. Suppose a random item is being searched for. What is the probability that it is in the bottom region and what is the probability that it is in the upper region.
   d. What is the average time of the modified algorithm? (This question may be time consuming. For part credit, just give the recurrence equation for the time.)

2. Suppose \( ue^{1+u^2} = x \). Find an approximation for \( x \) in terms of \( u \) that is good for \( u \) near zero. If you use asymptotic iteration, you may wish to use \( x = u \) as your initial guess.

3. Approximate \( \sum_{1 \leq i \leq n} \frac{1}{i^4} \).

4. Solve the recurrence \( T_n = T_{n-1} + 2T_{n-3} \).

5. Suppose you are designing a divide and conquer algorithm that you want to run in time less than \( n^4 \).
   a. If your overhead of the dividing part of the algorithm is \( n^k \) for some integer \( k \), what is the biggest permissible value for \( k \).
   b. If the parts are half the size of the original problem what is the maximum number of parts that you can have.