Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

**Problem 3:** Exercise 1.4.6: Give the order of growth (Big-O notation) as a function of n of the running times of each of the following code fragments.  Please justify your reasoning.

1. int sum = 0;  
   for (int k = n; k > 0; k /= 2)  
     for (int i = 0; i < k; i++)  
       sum++;

k = 5; k = 2; 2nlogn

sum = 5; sum = 7;

The growth order for this code fragment is 2nlog(n). The reason I say this is because the function will increment by this much between each step. I figured this out mathematically by entering numbers and finding the values that they grew to.

1. int sum = 0;  
   for (int i = 1; i < n; i \*= 2)  
     for (int j = 0; j < i; j++)  
       sum++;

i = 1; i = 2; i = 3; i = 4; 2n

sum = 1; sum = 3; sum = 6; sum = 10;

The growth order for this code fragment is 2n. I say this because the sum is growing in increments of double what the n value is.

1. int sum = 0;  
   for (int i = 1; i < n; i \*= 2)  
     for (int j = 0; j < n; j++)  
       sum++;

i = 1; i = 2; i = 3; i = 4; i = 5; n^2

sum = 5; sum = 10; sum = 15; sum = 20; sum = 25;

The growth order for this code fragment is n^2. I say this because the final value ends up being the square of the value you put in when all of the code has been gone through.