### Activity3

Submit in a word document or a text file.

Give tilde approximations for the following quantities

1. N + 1

Ans: ~N

2. 1 + 1/NAns: ~1

1. (1 + 1/N) (1 + 2/N)

Ans: ~1

1. 2N^3 - 15 N^2 + N

Ans: ~N

1. Lg (2N)/lg N

Ans: ~1/2

1. Lg (N^2 + 1) / lg N

Ans: ~0

### Activity4

Submit in a word document or in a text file

Give the order of growth (as a function of N ) of the running times of each of the following code fragments:

**Note:** use the following Notations for writing Answers: N, 1, log(N), N^2, N\*log(N), 2^N, N^3, log(N)

O(N):

The time complexity of a Loop is considered as O(N), if the variable is incremented or decremented.

**Code-1:**     int count = 0;

            for(int i = 0; i < N; i++)

{

                count++;

            }

O(1)

It does not contain any loop.

int sum = 0;

            if(sum == 0)

{

                sum++;

}

O(log(N))

Time complexity of loop is considered as O(logN), if the loop is divided /multiplied by constant amount

 for(int i = N; i > 0; i < N/2)

{

            int sum = 0;

}

O(N^2)

Time Complexity of the nested loop is equal to number of times inner most loop excited.

 for(int i = 0; i < N; i++){

                for(int j = 0; j < N; j++){

                    System.out.println(“Hello”);

                }

            }

O(N\*log(N))

Step1:The time complexity of a Loop is considered as O(N), if the variable is incremented or decremented.

Step2:Time complexity of loop is considered as O(logN), if the loop is divided /multiplied by constant amount

for(int i = 0; i < N; i++) // Step1

{

   for(int j = 0; j < N; j = j \* 2) // Step2

{

     System.out.println(“Hello”);

}

}

O(2^N)

2^N=n

N=log n.

for(int i = N; i > 0; i < N/2)

{

            int sum = 0;

}

O(N^3)

### Activity5

Submit in a word document or in a text file

Give the order of growth (as a function of N ) of the running times of each of the following code fragments:

**Note:** use the following Notations for writing Answers: N, 1, log(N), N^2, N\*log(N), 2^N, N^3, log(N)

* 1. O (N\*log(N))

Step1:The time complexity of a Loop is considered as O(N), if the variable is incremented or decremented.

Step2:Time complexity of loop is considered as O(logN), if the loop is divided /multiplied by constant amount

  int sum = 0;  
        for (int n = N; n > 0; n /= 2)  
            for(int i = 0; i < n; i++)  
                Sum++;