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)

Solutions:

**Code-1:**

int count = 0;

 for(int i = 0; i < N; i++)

{

                count++;

 }

**Time complexity : N - loop runs N times**

**Code-2:**

int sum = 0;

if(sum == 0)

{

                sum++;

}

**Time complexity: 1 – statement executes only one time**

**Code-3:**

 for(int i = N; i > 0; i < N/2)

{

            int sum = 0;

}

**Time complexity: logN – every time N value becomes N/2**

**Code-4:**

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

{

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

{

                    System.out.println(“Hello”);

                }

}

**Time complexity: N^2 – inner loop N times and outer loop N times N\*N=N^2**

**Code-5:**

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

{

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

{

                    System.out.println(“Hello”);

                }

}

**Time complexity: NlogN – inner loop takes logN times and outer loop takes N times**

**Code-6:**

public int fibonacci(int number)

{

if (number <= 1)

{

                     return number;

            }

else

{

                     return fibonacci(number - 1) + fibonacci(number - 2);

            }

}

**Time complexity: N^2**