**Written Problems:**

**1:**

def func(n):

j = n

while j >= 1:

for i in range(j):

val = func1(n)

j = j // 3

In the 1st while-loop, j = n2. for-loop executes Call.Func1(n) n times.

In the 2nd while-loop, j <= n2/3, n/3 times

In the 3rd while-loop, j <= n2/9, n/9 times

In the 4rd while-loop, j <= n2/27, n/27 times

…..

In the zth while-loop, j <= n2/3z n2/3z times = 1

g(n) = n + (n/3) + (n/9) + (n/27) + …. + (n2/3z) where z = log3 n2

**Geometric Sum Formula:**

a + ar + ar2 + …. + arm = a(rm+1 – 1)/(r – 1), where r != 1.

g(n) = n2 + n2/3 + n2/9 + … + n2/3z

g(n) = n2 + n2\*(1/3) + n2\*(1/3)2 + … + n2\*(1/3)z

g(n) = n2((1/3) z+1 -1)/(1/3 – 1)

g(n) = n2( (1/3) z+1-1)/(-2/3)

g(n) = -3n2/2(1/3z+1-1)

g(n) = ½\*n2(3 – 3\*1/3z+1)

g(n) = ½\*n2(3 – 1/3z)

g(n) = (3n2)/2 – n2/3z

g(n) = (3n2)/2 – 1

g(n) = (3n2)/2 – 1

let k = 3/2, n0 = 1

Big O is O(n2)

**2:**

def func2(n):

for i in range(1, n+1): n

for j in range(i,1 + (i\*n)): n\*(n\*n)

task(i+1,j) n\*(n2+1)

Big(O) of func2(n) = O(n3)

After eliminating the lower terms: Big(O) = O(n3)

g(n) = ∑ni=1 ∑i\*nj=1 1

g(n) = ∑ni=1 1(i\*n-i+1)

g(n) = ∑ni=1 i\*n-i+1

g(n) = n\*∑ni=1i - ∑ni=1 I + ∑ni=1 1

g(n) = n2/2(n+1)-n(n+1)/2+1(n-1+1)

g(n) = (n3+n2-n2-n+2n)/2

g(n) = (n3+n2)/2

Big O is O(n3)