f(n) = 1000 n

g(n) = 2 n^2

Find smallest n where:
g(n) >= f(n)

2.n^2 >= 1000 n
n >= 500

n	f(n)	g(n)	f(n) / f(n-1)	g(n) / g(n-1)
1	1000	2	-	-
2	2000	8	2	4
3	3000	18	1.5	2.25
4	4000	32	1.33	1.78
5	5000	50	1.2	1.5625
.....				

Scalability

1) f(n) is O(g(n)) => g(n) is Ω(f(n))

2) f(n) is Ω(g(n)) => g(n) is O(f(n))

3) f(n) is Θ(g(n)) <=> f(n) is O(g(n)) AND f(n) is Ω(g(n))

To prove: f(n) is Θ(g(n)) <=> g(n) is Θ(f(n))

f(n) is Θ(g(n)) => f(n) is O(g(n)) AND f(n) is Ω(g(n)) [apply rule 3]
=> g(n) is Ω(f(n)) AND g(n) is O(f(n)) [apply rules 1 and 2]
=> g(n) is Θ(f(n)) [apply rule 3]

---

f(n) = 2.n^2 + 4.n^3
= d(n) + e(n)
= O(n^2 + n^3)
= O(g(n))

g(n) = O(n^3)

Thus, f(n) = O(n^3)

d(n) = 2.n^2 is O(n^2)

e(n) = 4.n^3 is O(n^3)

g(n) = n^2 + n^3
Now, g(n) is a polynomial of degree 3,
g(n) is O(n^3).

---

f(n) = 3.log(n) + log(log(n))
= d(n) + e(n)
= O(log(n) + log(n))
= O(2.log(n))
= O(b(n))

b(n) = 2.log(n) is O(log(n))

Thus, f(n) is O(log(n))

d(n) = 3.log(n) is O(log(n))

e(n) = log(log(n)) is O(log(n))

So, e(n) is O(log(n))

2n^5 is O(n^5)

n^5 + 3 is O(n^5)