and $c_3 n$, respectively, then we know that the one with complexity $c_3 n$ will be faster than the one with complexity $c_1 n^2 + c_2 n$ for sufficiently large n.

- For small values of n, either program could be faster (depending on c_1, c_2 and c_3).
- If $c_1 = 1, c_2 = 2$ and $c_3 = 100$, then $c_1 n^2 + c_2 n \le c_3 n$ for $n \le 98$ and $c_1 n^2 + c_2 n > c_3 n$ for $n > 98$. If $c_1 = 1, c_2 = 2$ and $c_3 = 1000$, then $c_1 n^2 + c_2 n \le c_3 n$ for $n \le 998$.
- No matter what the values of c_1, c_2 and c_3, there will be an n beyond which the program with complexity $c_3 n$ will be faster than the one with complexity $c_1 n^2 + c_2 n$.

Let, $c_3 n > c_1 n^2 + c_2 n \Rightarrow \boxed{n(c_1 n + c_2 - c_3) < 0}$. If $n > 0$, $c_1 n + c_2 - c_3 < 0$

$\boxed{n < \frac{c_3 - c_2}{c_1}}$ → No upper bound on n.

If $n < 0$, $c_1 n + c_2 - c_3 > 0$, $\boxed{n > \frac{c_3 - c_2}{c_1}}$ → no lower bound on n

- This value of n will be called the $\boxed{\text{break even point}}$. If the break even point is 0 then the program with complexity $c_3 n$ is always faster (or atleast as fast).
- The exact break event point cannot be determined analytically. The programs have to be run on a computer in order to determine the break event point. (This is a practical epistemological claim, not a mathematical impossibility claim)

$\boxed{\text{Defn:}}$ [Big "oh"] $f(n) = O(g(n))$ (read as "f of n is big oh of g of n") iff there exist positive constants c and n_0 such that $\boxed{f(n) \le c\,g(n) \text{ for all } n, n \ge n_0}$ □
(f, g are non-negative functions)

$\boxed{\text{Ex-1.15:}}$ $3n + 2 = O(n)$ as $3n + 2 \le 4n$ for all $n \ge 2$. $3n + 3 = O(n)$ as $3n + 3 \le 4n$ for all $n \ge 3$. $100n + 6 \le 101n$ for $n \ge 10$. $10n^2 + 4n + 2 = O(n^2)$ as $10n^2 + 4n + 2 \le 11n^2$ for $n \ge 5$. $1000n^2 + 100n - 6 = O(n^2)$ as $1000n^2 + 100n - 6 \le 1001n^2$ for $n \ge 100$.
$6 * 2^n + n^2 = O(2^n)$ as $6 * 2^n + n^2 \le 7 * 2^n$ for $n \ge 4$.

- The statement $f(n) = O(g(n))$ only states that $g(n)$ is an upper bound on $f(n)$ for all $n, n \ge n_0$. It doesn't say anything about how good the bound is. $n = O(n^2)$, $n = O(n^{2.5})$, $n = O(n^3)$, $n = O(2^n)$. In order for the statement $f(n) = O(g(n))$ to be informative, $g(n)$ should be as small as a function of n one can come up with for which $f(n) = O(g(n))$.