Question 8.

Algorithms Assignment 1

Faaiq Bilal
23100104

1 Part 1

This can be disproved with a counter-example. If we take $f(n) = n$ and $g(n) = n^2$
Then the first condition is satisfied as $n < c \cdot n^2$ for all $n > 0$ and $c \geq 1$
However, the converse is not necessarily true, as n^2 grows faster than $c \cdot n$.
Hence, in this case, $g(n) \notin f(n)$

2 Part 2

The proof of this statement can be expressed in two parts:

2.1 Proof part 1, Big O

For this, we can start off by supposing that $f(n) > g(n)$. This supposition doesn't actually matter, it only helps us in denoting one function as being larger than the other. The roles could very easily have been swapped.

In such a case, the following must also be true for any constant $c_0 \geq 2$:
$c \cdot f(n) \geq f(n) + g(n)$

This proves that $f(n) + g(n) \; \epsilon \; max(f(n), g(n))$

2.2 Proof part 2, Big Omega

This part is much simpler to prove, as function $f(n)$ and $g(n)$ grow with increasing n.
This would mean that $f(n) + g(n)$ must be larger than both $f(n)$ and $g(n)$ individually. It does not matter which of them is larger.
Hence, we can say that $f(n) + g(n) \geq max(f(n), g(n))$
$f(n) + g(n) \; \epsilon \; max(f(n), g(n))$
$f(n) + g(n) \; \epsilon \; \Omega(max(f(n), g(n)))$

2.3 Conclusion

As both statements above are true, then $f(n) + g(n) \ \epsilon \ \theta(max(f(n) + g(n)))$

3 Part 3

This statement is true as shown below:

$f(n) = O(g(n))$ can also be written as: $f(n) \leq M \cdot g(n)$, where M is some constant > 0.
This statement can be extended as follows: $lg(f(n)) \leq lg(M \cdot g(n))$
Which will further mean that:
$lg(f(n)) \leq K \cdot (lgM + lg(g(n)))$ where K is some constant > 0