Assignment 3

problem 3.1

a) $f(n) = 9n$ and $g(n) = 5n^3$

$f \in \Theta(g)$: Yes, as $f(n) = \frac{9}{5}(g)$ thus grows at same rate

$f \in O(g)$: NO, $f(n)$ grows faster than $g(n)$

$f \in o(g)$: NO, $f(n) \rightarrow$ linear, $g(n) \rightarrow$ cubical (grows at same rate or slower than $g(n)$ but ~~must~~ does not strictly slower than $g(n)$)

$f \in \Omega(g)$: NO, $f(n)$ does not grow as fast as $g(n)$

$f \in \omega(g)$: NO, $f(n)$ doesn't grow strictly faster than $g(n)$

$g \in \Theta(f)$: no, $g(n)$ does not grow at the same rate of $f(n)$

$g \in O(f)$: no, because $g(n)$ is not slower than $f(n)$

$g \in o(f)$: Yes, $g(n)$ grows strictly slower than $f(n)$ because $g(n)$ grows cubicly while $f(n)$ grows logarithmically cubed

$g \in \Omega(f)$: NO, because $g(n)$ does not grow faster than $f(n)$

$g \in \omega(f)$: no, $g(n)$ does not grow strictly faster than $f(n)$

b) $f(n) = 9n^{0.8} + 2n^{0.3} + 14\log n$ and $g(n) = \sqrt{n}$

$f \in O(g) \rightarrow$ true, as $f(n)$ is upper bound of $g(n)$

for the rest all $\rightarrow$ False, as $f(n)$ and $g(n)$ have different rates of growth

c) $f(n) = \frac{n^2}{\log(n)}$ $g(n) = n\log n$

$f \in \Theta(g)$: False, as $n \to \infty$ $n\log n$ dominates $\frac{n^2}{\log(n)}$

Since $f \in \Theta(g)$ is false

↳ ALL FALSE $f \in \omega(g)$, $g \in \Theta(f)$, $g \in O(f)$, $g \in \Omega(f)$, $g \in \omega(f)$

$f \in O(g)$: True, $f(n)$ is upper bound of $g(n)$ as proved in $f \in \Theta(g)$

$f \in o(g)$: False, $f(n)$ grows strictly slower than $g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

$f \in \Omega(g)$: True, $f(n)$ is a lower bound: $f(n)$ grows at least as fast as $g(n)$

d) $f(n) = (\log(3n))^3$ and $g(n) = 9\log n$

TRUE

$f \in O(g)$ → as similar to part(a)

$f \in \Theta(g)$

Problem 3.2

a) SELECTION-SORT(A)

```
n ← length(A)
for (i = 1 to n)
    imin ← i
    for (j = i+1 to n)
        min ← j
    swap(A[i], A[min])
```

b) Loop invariant: At the start of each iteration of the outer loop, the subarray A[1..i-1] contains the i-1 smallest elements of A in sorted order

c) Lets say n = 20

Case A: Lets say we want them to have the most swaps
For most swaps the array must generate random order elements in

case B: For least swaps the array that is generated must be already sorted hence no swaping done.

Explained to .py file

e) Average case : As n increases, the time taken for selection sort grows quadratically $O(n^2)$

Case A : Takes the longest time due to max number of swaps
- Grape similar to Average case but steeper

Case B :
- input already sorted, thus shortest time
- curve is flattest among compared to Average case and case A