HOMEWORK #2

(Q1)

Q1.a) $\lim_{n\to\infty} \frac{(n^2-3n)^2}{5n^3+n} = \frac{n^4-6n^3+9n^2}{5n^3+n}$

$\xrightarrow{\text{L'hospital}} \lim_{n\to\infty} \frac{4n^3-18n^2+18n}{15n^2+1}$

$\xrightarrow{\text{L'hospital}} \lim_{n\to\infty} \frac{12n^2-36n+18}{30n}$

$\xrightarrow{\text{L'hospital}} \lim_{n\to\infty} \frac{24n-36}{30} \to \infty$

Therefore $f(n) \in \Omega(g(n))$

Q1.b) $\lim_{n\to\infty} \frac{n^3}{\log_2 n^4}$

$\xrightarrow{\text{L'hospital}} \frac{3n^2}{\frac{4}{n\ln 2}} = \frac{\ln(2)\ 3n^3}{4} = \infty$

Therefore $f(n) \in \Omega(g(n))$

Q1.c) $\lim_{n\to\infty} \frac{5n\log_2(4n)}{n\log_2(5^n)} = \frac{5n\log_2 4n}{n^2\log_2 5} = \frac{5\log_2 4n}{n\log_2 5}$

$\xrightarrow{\text{L'hospital}} \frac{\frac{5}{\ln(2)n}}{\log_2 5} = \frac{5}{\log_2 5 \cdot \ln(2) \cdot n} = \frac{5}{\infty} = 0$

Therefore $f(n) \in O(g(n))$

Q1.d) $\lim_{n\to\infty} \frac{n^n}{10^n} = \left(\frac{n}{10}\right)^n$

$\lim_{n\to\infty} \left(\frac{\infty}{10}\right)^{\infty} = \infty$

Therefore $f(n) \in \Omega(g(n))$

Q1.e) $\lim_{n\to\infty} \frac{\ln \sqrt[5]{2n}}{n\sqrt[3]{n}} = \frac{8 \cdot 2^{1/5} \cdot n^{4/5}}{n^{4/3}} = \frac{8 \cdot 2^{1/5} \cdot n^{-\frac{2}{15}}}{1}$

$\lim_{n\to\infty} \frac{8 \cdot 2^{1/5}}{n^{2/15}} = \frac{8 \cdot 2^{1/5}}{\infty} = 0$

Therefore $f(n) \in O(g(n))$

Q2.a) methodA iterates each element of str-array. So it will run 'n' times.
Therefore worst case time complexity is $O(n)$

Q2.b) Since methodA is called inside a for loop that runs 'n' times and methodA is also running 'n' times, time complexity of the first two line is $O(n^2)$.
The second loop will run n times therefore it is $O(n)$.
So overall worst case time complexity is $O(n^2+n) = O(n^2)$

Q2.c) there are nested loops in the first two lines, which makes $O(n^2)$ time complexity. But inside this nested loop, methodB is called which we know its complexity is $O(n^2)$. So, overall time complexity is $O(n^4)$

Q2.d) Since the loop runs infinetely, there is no possible complexity analysis for this algorithm.

Q2.e) In the worst case, the empty string would be at the last position of the string. That means the loop have to run 'n' times till it can break and that makes worst case time complexity $O(n)$.

Q3.a) • maxDiff(A)

return A[n-1] - A[0]

• In an ascending array, the last element becomes the largest while first one becomes the smallest. Therefore we just need to subtract first from last. The worst case time complexity is O(1) because accessing elements in an array is constant time.

Q3.b)

• maxDiff(A)

min = A[0]

max = -9999 // max should a very small integer

for i from 0 to n-1

if a_i > max

max = a_i

if a_i < min

min = a_i

return max - min

• Loop iterates trough each element and it finds the min and max by comparing the current element with the variables 'min' and 'max'. At the end, it simply returns the difference between max and min. Worst case time complexity is O(n) because the loop iterates n times.