Assignment 2 Problem 1: Divide & Conquer

a) Since for each a[i] where i is in [1 ⋯ n], $\lg a[i] \in O(\log a[i])$. And since a[i] is a primitive type integer, therefore, $\lg a[i] \in O(64) = O(1)$

For $P \coloneqq Mul(P,\ LongInt(a[i]))$, the runtime should be $O\left(\max(\lg P, O(1)) \cdot \min(\lg P,\ \mathrm{O}(1))^{\log_2 3 - 1}\right) = O(\lg P)$

Focus on P, in each iteration, P is multiplied by a primitive type integer, therefore, each time P increases, $P \cdot a[i] < 2^{64} \cdot P$, and $O(\lg P_{new}) = O(2^{64} \cdot \lg P_{old}) = O(64) + O(\lg P_{old})$. Since P is multiplied by n-1 times, and originally, $P = a[1] \in O(64) = O(1)$, the final value $O(\lg P) = O(n)$
Therefore, the total running time for the algorithm should be

$$\sum_{i=2}^{n} O(64i) \in O\left(\sum_{i=1}^{n} O(i)\right) \in O(n^2)$$

b) The algorithm did the following thing:
The algorithm firstly separates the array a into two equal length part and recursively call itself on each part. And apply $Mul(A, B)$ on the returning value of the two parts. And the base case of the recursion is if the length of a is 1, then apply $LongInt()$ on that entry.
The pseudo code is as follow:

$MyMultiMul(a[1 \dots n], n)$
$\quad if\ n = 1\ then$
$\quad\quad return\ LogInt(a[1])$
$\quad mid = n/2$
$\quad A = MyMultiMul(a[1 \dots mid], mid)$
$\quad B = MyMultiMul(a[mid + 1 \dots n], n - mid)$
$\quad return\ Mul(A, B)$

The base case of the recursion is O(1), and for each recursive call it's obviously that

$$T(n) = 2T\left(\frac{n}{2}\right) + O\left(n \cdot n^{\log_2 3 - 1}\right) = 2T\left(\frac{n}{2}\right) + O(n^{\log_2 3})$$

Note: from part a) we have already show that $O(\lg P) = O(n)$ which also applies to the number of bits for returning values of each parts.
Applying Master Theorem with $a = 2, b = 2, k = \log_2 3$, we have
$2 = a < b^k = 2^{\log_2 3} = 3$, and $T(n) = \Theta\left(n^{\log_2 3}\right) \in O(n^{\log_2 3})$