1.

Given the public key (7, 143) $N = 143$, $e = 7$
First found 2 prime numbers multiplied that equal, 143. In this case 11 and 13.
Then had to find ϕ, which is $(11-1)(13-1) = 120, \phi = 120$
Given that the inverse is between 100 and 105, I narrowed it down to 101 and 103 as they are the only prime numbers out of those.
$101 * e = 707 \Rightarrow 707 \bmod 120 = 107$, 101 is not an inverse.
$103 * e = 712 \Rightarrow 712 \bmod 120 = 1$, 103 is an inverse.
So now we have $N = 143$, $e = 7$, $phi = 120$, $d = 103$ Cipher or C = 106
decryption $= C^d \bmod 143 = 106^103 \bmod 143 = 24$
Therefore the decrypted message is 24.

2.

a)

yes, the function performs at $O(nlog(n))$ or $O(n^3)$ which $O(nlog(n))$ still fits into the ceiling of $O(nlog(n))$

b)

no, the function performs at $\Omega(nlog(n))$ or $\Omega(n^3)$ which both do not satisfy $\Omega(nlog(n))$ therefore it is no

c)

no, because the function performs at $O(nlog(n))$ or $O(n^3)$ which $O(nlog(n))$ does is not $O(n^3)$ as it's below that ceiling

d)

yes, the function performs $\Omega(nlog(n))$ or $\Omega(n^3)$ which both fit into the floor of $\Omega(n^3)$, which satisfies both

3.

a)

arithmetic	s	m
m = a[0]	undefined	6
s = 0	0	6
s = s + a[j]	6	6
s = s + a[j]	-1	6
s = s + a[j]	7	6
m = s	7	7
s = s + a[j]	10	7
m = s	10	10
s = s + a[j]	19	10
m = s	19	19
s = s + a[j]	7	19
s = 0	0	19
s = s + a[j]	-7	19
s = s + a[j]	1	19
s = s + a[j]	4	19
s = s + a[j]	13	19
s = s + a[j]	1	19
s = 0	0	19
s = s + a[j]	8	19
s = s + a[j]	11	19
s = s + a[j]	20	19
m = s	20	20
s = s + a[j]	8	20
s = 0	0	20
s = s + a[j]	3	20
s = s + a[j]	11	20
s = s + a[j]	-1	20
s = 0	0	20
s = s + a[j]	9	20
s = s + a[j]	-3	20
s = 0	0	20
s = s + a[j]	-12	20

final value of m = 20

b)

21 times

c)

algorithm	Θ
m = a[0]	1
i loop	n
s = 0	$n * 1$
j loop	$n * (n - i)$
s = s + a[j]	$n * (n - i) * 1$
s ¿ m	$n * (n - i) * 1$
m = s	$n * (n - i) * 1$

$\Theta(n^2)$, shown by the table above. J loop will happen n-i times which also includes everything, inside of it will also happen n-i times. We then multiply the individual Θ times with the times it happens according to the loop they are in. We will then add all the Θ times together. From there we can simplify to the highest Θ, which in this case is $n * (n - i) * 1$ which then can be simplified to just $\Theta(n^2)$.

4.

a)

algorithm	s	m
m = a[0]	undefined	6
s = 0	0	6
s = s + a[i]	6	6
s = s + a[i]	-7	6
s = 0	0	6
s = s + a[i]	8	6
m = s	8	8
s = s + a[i]	11	8
m = s	11	11
s = s + a[i]	20	8
m = s	20	20
s = s + a[i]	8	20

final value of m = 20

b)

algorithm	Θ
m = a[0]	1
i loop	n
s = 0	$n * 1$
s = s + a[j]	$n * 1$
s > m	$n * 1$
m = s	$n * 1$
s < 0	$n * 1$
s = 0	$n * 1$

$\Theta(n)$, shown by the table above. Similar to problem 3, m = a[0] will happen at $\Theta(1)$, everything inside the loop will happen n times their theta, which in this case everything inside performs at $\Theta(1)$. This results in the highest Θ to be $n * 1$ which can be simplified to $\Theta(n)$