(b) Prove that the amortized time complexity of your enqueue and dequeue operations is O(1)

Let's say we have the dynamic array of size (n).
now for (n) insertion, it takes O(1) time.
For inserting $(n+1)^{th}$ element, we need to grow the queue {enhance size of queue}.
⇒ new dynamic array of some (factor * n) size created.
(Generally this factor = 2).
Now copy the n elements ⇒ $T(1)*n$ time.
inserting $(n+1)^{th}$ element = $T(1)$ time.

⇒ total time for $(n+1)^{th}$ insertion = $T(1)*n + T(1) \simeq T(1)*(n+1)$
time complexity for $(n+1)^{th}$ insertion = $O(n)$.

For all (n+1) insertion, time taken = $\underbrace{T(1)*n}_{\text{for } n \text{ insert}} + \underbrace{T(n)}_{\text{for } (n+1)^{th} \text{ insert}} = T(1)*2n$

⇒ ammortized time complexity for
enqueue = $\frac{T(1)*(n+1)}{n+1} = O(1)$.

Similarly for dequeue,
till the size of array do not become very small
{capacity / (factor)²} → compared to capacity of array, we can perform dequeue operation in O(1) time. But when we have to shrink the array, which means making a dummy array & copying elements to that & then perform dequeue.
with generally factor = 2

⇒ Total time for (n+1) dequeue = $\underbrace{T(1)*n}_{\text{for basic } n \text{ dequeue}} + \underbrace{n*T(1)}_{\text{for } (n+1)^{th} \text{ dequeue}} = 2nT(1)$

for all (n+1) dequeue, time taken =

ammortized time complexity = $\frac{2nT(n)}{n+1} = O(1)$.