A Random Access Queue supports the operations `pushright(x)` to add a new item `x` to the tail, `popleft()` to remove the item at the head, and `element_at(i)` to retrieve the item at position `i` without removing it: `i = 0` gives the item at the head, `i = 1` the following element, and so on.

(a) We can implement this data structure using a simple linked list, where `element_at(i)` iterates from the head of the list until it reaches position `i`.

(i) State the complexity of each of the three operations. [1 mark]

(ii) A colleague suggests that, by defining a clever potential function, it might be possible to show that all operations have amortized cost $O(1)$. Show carefully that your colleague is mistaken. [6 marks]

(b) We can also implement this data structure using an array. The picture below shows a queue holding 4 items, stored within an array of size 8. When new items are pushed, it may be necessary to create a new array and copy the queue into it. The cost of creating an array of size `n` is $\Theta(n)$.

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\hline
\hline
\hline
\end{array}
\]

(i) Give pseudocode for the three operations. Each operation should have amortized cost $O(1)$. [6 marks]

(ii) Prove that the amortized costs of your operations are indeed $O(1)$. [7 marks]