In this problem set, we explore a well-known trick that uses two Stacks $S_1$ and $S_2$ to simulate the operations of a Queue $Q$. Consider the following implementations of `ENQUEUE` and `DEQUEUE`.

```plaintext
ENQUEUE(Q, x):
  if Size($S_1$) ≥ 12 and isEmpty($S_2$):
    while not isEmpty($S_1$):
      Push($S_2$, Pop($S_1$))
  Push($S_1$, x)

DEQUEUE(Q):
  if isEmpty($S_2$):
    if isEmpty($S_1$):
      error "Dequeueing from an empty queue!"
    else:
      while not isEmpty($S_1$):
        Push($S_2$, Pop($S_1$))
  return Pop($S_2$)
```

For your analysis, assume that each `Push` operation takes 2 units of time, and each `Pop` operation takes 3 units of time. Each `isEmpty` or `Size` operation takes 0 unit of time. The time taken by any other pseudo-code operation is ignored.

(a) Consider the sequence of operations that consists of 50 `ENQUEUE` operations followed by 50 `DEQUEUE` operations. Use aggregate analysis to compute the amortized cost per operation for this sequence.

(b) Now consider any sequence of $m$ `ENQUEUE` and `DEQUEUE` operations. Use the accounting method to derive an upper-bound on the amortized cost per operation.