Exercise 1: Amortized Analysis (2+4 points)

(a) Consider an extension of the augmented stack data structure from the lecture: In addition to offering a multipop($k$) operation, assume that the extended augmented stack also offers a multipush($k, L$) operation which takes a parameter $k \geq 1$ and a list $L$ of $k$ elements and it pushes each of these elements on the stack. Assume that both operations multipop($k$) and multipush($k, L$) require time $\Theta(k)$ to complete. In the lecture, we proved that for the simple augmented stack, the amortized cost of all operations is $O(1)$. Does this $O(1)$ the amortized cost bound for all stack operations continue to hold for the extended version of the augmented stack? Explain your answer!

(b) Show how to implement a queue with two ordinary stacks so that the amortized cost of each enqueue and dequeue operations is $O(1)$. Explain your amortized analysis.

Exercise 2: Fibonacci Heaps (2+4 points)

(a) Consider the following Fibonacci heap (black nodes are marked, white nodes are unmarked). How does the given Fibonacci heap look after a decrease-key($v, 2$) operation and how does it look after a subsequent delete-min operation?

(b) Fibonacci heaps are only efficient in an amortized sense. The time to execute a single, individual operation can be large. Show that in the worst case, both the delete-min and the decrease-key operations can require time $\Omega(n)$ (for any heap size $n$).

Hint: Describe an execution in which there is a delete-min operation that requires linear time and describe an execution in which there is a decrease-key operation that requires linear time.