1. Exercise 17.1-1 on page 456
   It does not hold. Assume that a stack is initially empty. Simply, a sequence of $n$ operations consisting of *MULTIPUSH*s and *MULTIPOP*s with $k$ needs $\Theta(nk)$ time, which yields the amortized cost of $\Theta(k)$.

2. Exercise 17.2-1 on page 458
   We can assign the following amortized costs to each stack operation.

   \[
   \begin{array}{|c|c|c|}
   \hline
   \text{Operations} & \text{Actual cost} & \text{Amortized cost} \\
   \hline
   \text{PUSH} & 1 & 2 \\
   \text{POP} & 1 & 2 \\
   \text{COPY} & \text{at most } k & 0 \\
   \hline
   \end{array}
   \]

   Figure 1: Actual costs and amortized costs for each stack operation

   We use one credit for each PUSH(POP) operation to pay the actual cost of the given operation and save the other credit. After $k$ operations, we have saved $k$ credits, which can be used to pay the cost of COPY operation. Due to the fact that the amortized costs above are constant, the cost of $n$ stack operations is $O(n)$.

3. Exercise 22.2-7 on page 602
   We create a graph by representing the wrestlers as vertices and the rivalries as edges. Then, choose one vertex at random. Let this vertex be the root and a "good guy" (It can be a "bad guy". It’s your choice). We run the breadth-first search (BFS) from the root and keep updating their distances from the root. All vertices with odd distances will be "bad guys" and with even distances will be "good guys". If you find two adjacent vertices being allocated to the same group, then return false. The time complexity of this algorithm is $O(n + r)$, which is the same as that of BFS.

3. Exercise 22.3-8 on page 611
   Here is a counterexample. Run DFS from vertex $a$.

   ![Counterexample](image)

   Figure 2: Counterexample