CPS 130 Homework 16 - Solutions

1. (CLRS 17.2-1) A sequence of stack operations is performed on a stack whose size never exceeds k. After every k operations, a copy of the entire stack is made for backup purposes. Show that the cost of n stack operations, including copying the stack, is $O(n)$ by assigning suitable amortized costs to the various stack operations.

   Solution: Assign the following amortized costs:

PUSH	3
POP	1
MULTIPOP	1

   PUSH uses one credit to pay for itself and saves one credit for future pops and one for copying the stack. POP and MULTIPOP pay for their operations using saved PUSH credits and save a credit for stack copying. After k operations, we have saved k credits exclusively for stack copying and can copy the stack for free. Since each operation costs at most $O(1)$ amortized and the credits are nonnegative, the cost for n operations is $O(n)$.

2. A sequence of n operations is performed on a data structure. The ith operation costs i if i is a power of 2, and 1 otherwise. Using the accounting method, determine the amortized cost per operation.

   Solution: Charge three credits for every operation. The amortized cost per operation is now $O(1)$. We show that this will always leave us with a non-negative amount of credits, even after expensive operations. It is clear that if i is not a power of 2, one credit pays for the operation and two credits are saved. By induction we will show if we can pay for $n = 2^k$ operations without running into debt, we can pay for $n = 2^{k+1}$ operations.

   Induction: For the base case, we can pay for $n = 1 = 2^0$ operations since we charge the first operation three and it only costs one. Assume we can pay for $n = 2^k$ operations for some k. Then the cost needed to pay for $n = 2^{k+1}$ operations is $C_1+C_2+C_3$. C_1 is the cost for operation 2^{k+1} which is 2^{k+1}. C_2 is the cost of the operations $2^k + 1$ through $2^{k+1} - 1$. Since each of these operations cost one and there are $2^k - 1$ such ops, $C_2 = 2^k - 1$. C_3 is the cost of the first 2^k operations. By our inductive hypothesis, C_3 is paid for. We must pay $C_1 + C_2 = 2^{k+1} + 2^k - 1$ for the remaining operations. The 2^k operations between 2^k and 2^{k+1} save $3 \cdot 2^k = 2 \cdot 2^k + 2^k = 2^{k+1} + 2^k > 2^{k+1} + 2^k - 1$ credits which can pay the necessary costs and have one credit left over.