1) (10 pts) ANL (Algorithm Analysis)

Consider storing a table with indexes 0 to N-1, where N = k^2, for some positive integer k, that starts with all entries equal to 0 and allows two types of operations: (1) adding some value to a particular index, and (2) querying the sum of all the values in the table from index 0 through index *m*, for any positive integer *m* < N. One way to implement a "table" to handle these two operations is to store two separate arrays, *groups,* of size k and *freq,* of size N. *freq stores* the current value of each index in the table. For the array *groups*, index i ($0 \le i < k$) stores the sum of the values in *freq* from index ik to index (i+1)k-1. (For example, if N = 25, then *groups[2]* stores the sum of the values of freq, from *freq[10]* through *freq[14]*, inclusive.

Determine, ***with proof***, the run-time of implementing operation (1) on this table using this storage mechanism and determine, ***with proof***, the run-time of implementing operation (2) on this table using this storage mechanism. (For example, if N = 100 and we had a query with *m* = 67, to get our answer we would add *groups[0], groups[1], groups[2], groups[3], groups[4], groups[5], freq[60], freq[61], freq[62], freq[63], freq[64], freq[65], freq[66]* and *freq[67]*. Notice that since the ranges 0-9, 10-19, 20-29, 30-39, 40-49, and 50-59 are fully covered in our query, we could just use the groups array for each of those sums. We only had to access the freq array for the individual elements in the 60s.)

Your answers should be Big-Oh answers in terms of N as defined above.