Consider a hash table implemented with open addressing. This question is concerned with how its underlying storage array is to be managed in order to efficiently support a dynamic collection of items—in particular, when it is to be resized and its contents rehashed.

Let the underlying storage array have capacity $c$, and let it consist of $n$ slots containing values, and $d$ slots with a DELETED marker, plus empty slots. For performance reasons we wish to maintain $n + d \leq \gamma c$, where $\gamma = 0.75$ is the maximum allowed load factor. The set and delete operations then take $O(1)$ time to update the array contents; they might also trigger a rehash which takes $O(n + d)$.

(a) Define potential function. Given a potential function, explain how one can use it to obtain amortized costs. [3 marks]

(b) When should the rehashing be triggered, and what should the new size be? Give a policy for which set and delete have amortized cost $O(1)$. Using a potential function, or otherwise, justify your answer. [7 marks]

For memory efficiency reasons we don’t want to store too many DELETED markers: we wish to maintain $c \leq (1 + \delta)n$, where $\delta = 3$ is the maximum allowed memory overhead.

(c) Modify your procedure (if necessary) to support this memory constraint in addition to the load factor constraint. Justify why set and delete have amortized cost $O(1)$. [Hint: Suppose the table is rehashed, and then there is a sequence of deletions, triggering a second rehash. What’s the smallest number of deletions needed to trigger the second rehash?] [10 marks]