Problem

Given a linked list where each node has a ***next pointer and a random pointer*** construct a ***deep*** copy of the linked list and return the head of the deep copied linked list.

---

1. Understanding the difference between ***shallow copy vs deep copy.***

My understanding of a shallow copy is that two types of collections share a reference or data within memory. Therefore if one of the copies gets changed then the other copy will change as well. Very little duplication occurs during a shallow copy.

As for deep copies, we must duplicate and clone an entire new copy. Therefore the two copies DO NOT share the same reference (area in location), so that when one copy faces change, the other copy will not be affected.

A good reference:
StackOverflow Answer

---

2. Approach #1 - Space Complexity O(n) / Time Complexity O(n)

Iteration #1: Iterating until we reach the end of the linked list.

> Create a hashmap of the {key=original_node, value=new_node}. A mapping between the original node and the cloned node, ignoring for now where next/random pointers are pointing to.

Iteration #2: Iterating from head to the end of the linked list again.

> Now that you have a mapping between the clones and the original nodes, simply iterate through the list again and attach the clones exactly how it's attached from the original using the map you created during your first iteration.

A good reference:
Youtube Video
