Random pointer

Okay, let's explain this step by step in a very easy way.

We are making a deep copy of a linked list where each node has two pointers:

- `next` → goes to the next node
- `random` → can point to any random node in the list

We want a new cloned list with the same structure (but new nodes).

Original List

```
1   2   3
```

Random links:

- `1.random = 3`
- `2.random = 1`
- `3.random = 2`

✅ Step 1: Interleave cloned nodes

For each original node, create a copy node (`1'`, `2'`, `3'`) and put it right after the original.

So it becomes:

```matlab
1   1'   2   2'   3   3'
```

Right now:

- Original nodes still have correct `random`
- Clones (`1'`, `2'`, `3'`) don't have `random` yet

✅ Step 2: Fix `random` for cloned nodes

Now, for each cloned node (`x'`),
we set its `random` to the clone of the original's random.

How do we get that easily?

- If `1.random = 3`,
  then `1'.random = 1.random.next` → `3'`
- If `2.random = 1`,
  then `2'.random = 2.random.next` → `1'`
- If `3.random = 2`,
  then `3'.random = 3.random.next` → `2'`

So now cloned nodes have correct random links:

```matlab
1'.random = 3'
2'.random = 1'
3'.random = 2'
```

✅ Step 3: Split the two lists

Finally, separate the two lists:

- Take out all originals → `1   2   3`
- Take out all clones → `1'   2'   3'`

Now we have a deep copy (`1'   2'   3'`)
with the same `random` structure as the original.

Why does this work?

- Interleaving makes it easy to find the clone of any random target (just `original.random.next`).
- After fixing all `random` pointers, we can safely split them.

Visual Summary

```matlab
Original: 1 --next--> 2 --next--> 3
          \random    \random    \random
           v          v          v
           3          1          2

After interleaving:
1    1'    2    2'    3    3'
|    |
v    v
3    3'

Fix random for clones:
1'.random = 3'
2'.random = 1'
3'.random = 2'

Split    done!
```

Would you like me to draw an ASCII diagram showing how random pointers work before & after cloning?