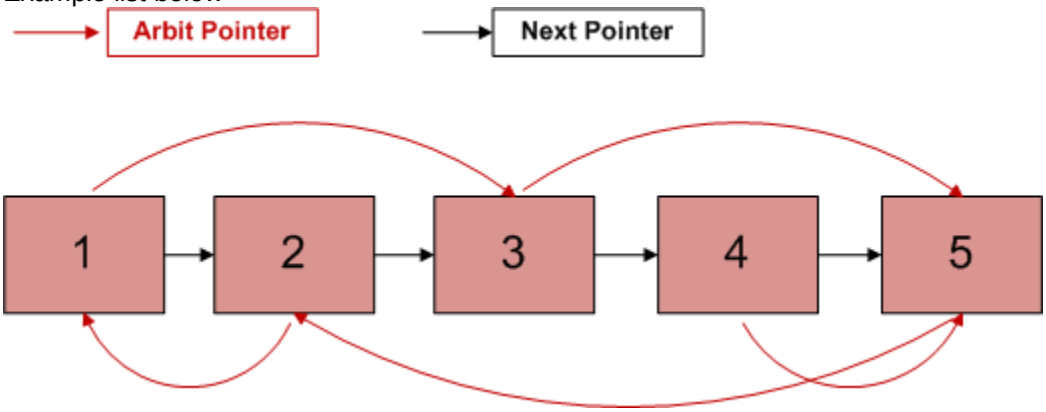
You are given a Doubly-Linked List with one pointer of each node pointing to the next node just like in a Singly-Linked list. The second pointer however CAN point to *ANY* arbitrary node in the list, not just the previous node. Now write a program/algorithm in **O(n)** time to duplicate this list.

The duplicated list must have both the values of the original list, as well as a duplication of the pointer-pathway-mapping of the original. "Next pointers" will point to the next node in the list, while the "Arbit pointer" of the list will point to *any* existing node in the list.

Pseudo-code or a detailed description of your algorithm is fine. Proof of O(n) time complexity is required.

No hashmap or additional memory is needed. The only memory allocated is what is needed for the new list. So, if the original linked list has *n* elements in the list, you may only allocate the *n* elements required to duplicate the list.

Example list below



If needed, the following structure definition in C may be used for the nodes of the double linked list:

```
typedef struct Node {
int value;
struct Node *next;
struct Node *arbit;
}
```