Remove duplicates from an unsorted linked list

Given an unsorted linked list of **N** nodes. The task is to remove duplicate elements from this unsorted Linked List. When a value appears in multiple nodes, the node which appeared first should be kept, all others duplicates are to be removed.

Example 1:

```
Input: N = 4
value[] = {5,2,2,4}
Output: 5 2 4 Explanation:Given linked list elements are
5->2->2->4, in which 2 is repeated only.
So, we will delete the extra repeated
elements 2 from the linked list and the
resultant linked list will contain 5->2->4
```

Example 2:

```
Input: N = 5
value[] = {2,2,2,2,2}
Output: 2 Explanation:Given linked list elements are
2->2->2->2->2, in which 2 is repeated. So,
we will delete the extra repeated elements
2 from the linked list and the resultant
linked list will contain only 2.
```

Your Task:
You have to complete the method **removeDuplicates**() which takes **1** argument: the **head** of the linked list. Your function should return a pointer to a linked list with no duplicate element.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(N)

Constraints:

1 <= size of linked lists <= 10^6

0 <= numbers in list <= 10^4

```
def removeDuplicates(self, head):
        # code here
        # return head after editing list
        if head is None or head.next is None:
            return head
```

```
        dummyHead = Node(-1)
        subset = set()
        curr = head
        prev = dummyHead
        while curr!=None:
            if curr.data in subset:
                curr = curr.next
            else:
                prev.next = curr
                prev = curr
                subset.add(curr.data)
                curr = curr.next
        if prev.next!=None:
            prev.next = None
        return dummyHead.next
```