# COPY LINKED LIST WITH RANDOM

☆ In this problem, we are given a linked list where each node is having an extra random pointer which points to a random Node in the list.

Our job is to create a deepcopy of this linked list and to return its head.

Bruteforce solution involves the use of a hashmap to remember nodes

C++ :

```
Node * copyRandomPointers(Node * head) {
    Node * t = head;
    unordered_map<Node *, Node *> mpp;
    while(t != nullptr) {
        Node* n = new Node(t→val);
        mpp[n] = t;
        t = t→next;
    }
    t = head;
    while(t != nullptr){
        Node* c = mpp[t];
         c→next = mpp[t→next];
         c→random = mpp[t→random];
         t = t→next;
    }
    return mpp[head];
}
```

Time Complexity is O(2N) and Space

Complexity is O(N).

Optimal Solution involves a few steps:
① Insert copy nodes in between real nodes
② Fix the random pointers for copy nodes
③ Remove the OG Nodes from this list.

C++ :

```
Node * copyRandomPointers(Node * head) {
   Node * temp = head;
   while(temp != nullptr) {
      Node * c = new Node(temp→val);
       c→next = temp→next;
      temp→next = c;
   }
   temp = head;
   while(temp != nullptr) {
      Node * c = temp→next;
       c→random = temp→random→next;
       temp = temp→next→next;
   }
    Node * dN = new Node(-1);
    Node * res = dN;
    temp = head;
    while(temp != nullptr) {
       res→next = temp→next;
       temp→next = temp→next→next;
        res = res→next;
        temp = temp→next;
    }
    return dN -> next ;
}
```