STARTING POINT OF A LOOP IN A LINKED LIST

☆ In this problem, we are given a singly linked list and we are supposed to return the node at which a loop starts in a Linked List.

Bruteforce Solution is to use a hashmap to store each node we go across, and check at each iteration if we stored this node already or not. If yes we just return that node otherwise return nullptr.

Optimal Solution again involves tortoise and hair algorithm. We take a slow pointer and a fast pointer, waiting for them to converge. When they do, we just return the node which started this loop.

```
Node*   detectLoop(Node * head) {
    Node * s = head ;
    Node * f = head ;
    while(f != nullptr && f->next != nullptr){
        s = s -> next ;
        f = f -> next -> next;
        if (s == f) {
            break ;
        }
    }
    if (f == nullptr || f->next == nullptr) {
        return nullptr ;
    }
    f = head ;
    while (f != s) {
        f = f->next ;
        s = s->next ;
    }
    return slow ;
}
```

We perform the last step to redirect ourselves to the origin of this loop. Fast pointer travels from head while slow pointer travels from within the loop. Both of them collide at the start of the loop