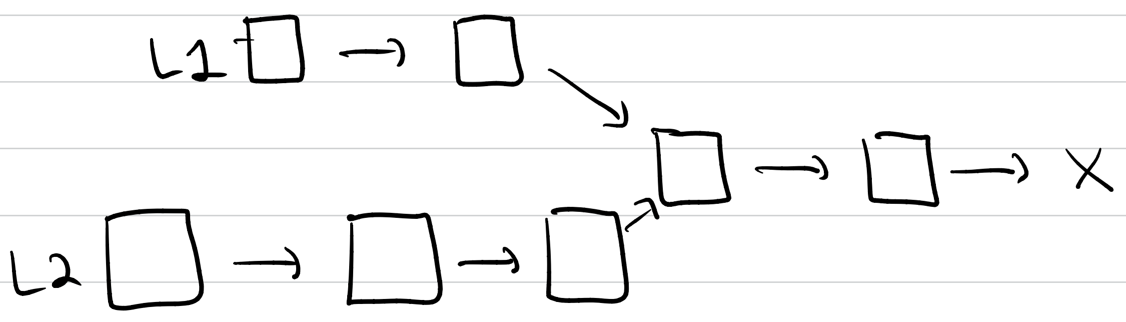
INTERSECTION POINT OF Y LINKED LIST

☆ In this problem we are given two linked lists which may or may not intersect. Eg :



Our job is to return the node where this intersection happens.

Brute force solution is to use a map to fill all the nodes we went through via L1. Then we just traverse the linked list via L2 and if a node exists in the map, that is our intersection point.

Better Solution is to have side by side comparisions during traversal. We do a length calculation for both lists, we check which list is longer, bring its head upto par with the shorter linked list and then side by side comparisions

Optimal Solution will not need any length calculation. We will just move each mover node 1 step at each time. When one mover reaches null it restarts from head of other linked list. Since there will be a difference in lengths, after N1 iterations L1 will be at L2 and after N2 iterations L2 will be at L2. The length difference is now neutralized and hence they are bound to collide.

C++ :

```
Node*  intersectionPoint(Node* h1, Node*
                                    h2) {
    if(h1 == nullptr || h2 == nullptr) {
        return nullptr;
    }
    Node* t1 = h1;
    Node* t2 = h2;
    while(t1 != t2) {
        t1 = t1 -> next;
        t2 = t2 -> next;
        if(t1 == t2) {
            return t1;
        }
        if(t1 == nullptr) {
            t1 = h2;
        }
        if(t2 == nullptr) {
            t2 = h1;
            }
        }
        return t1 ;
    }
```