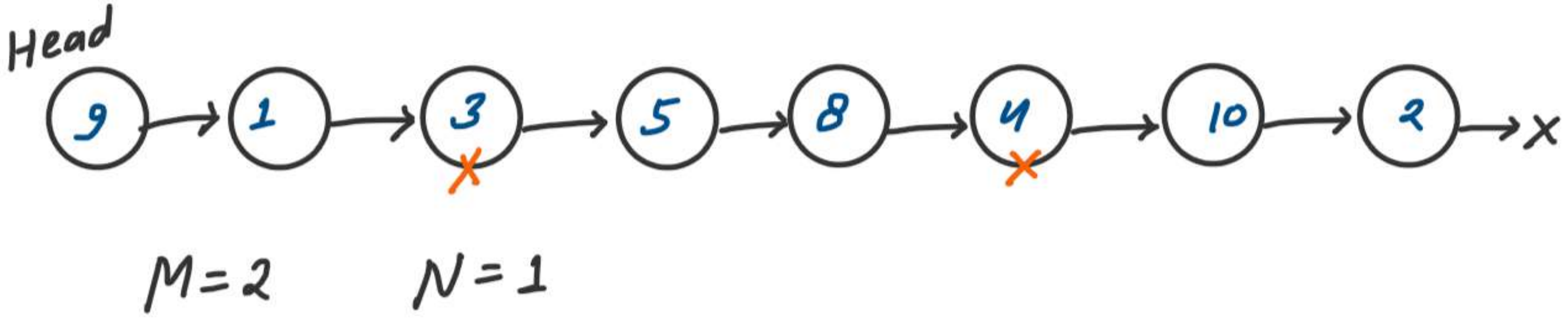
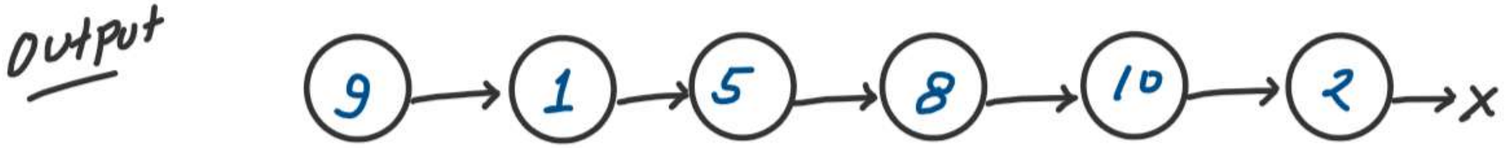
HW 04: Delete N Nodes after M Nodes (GFG)

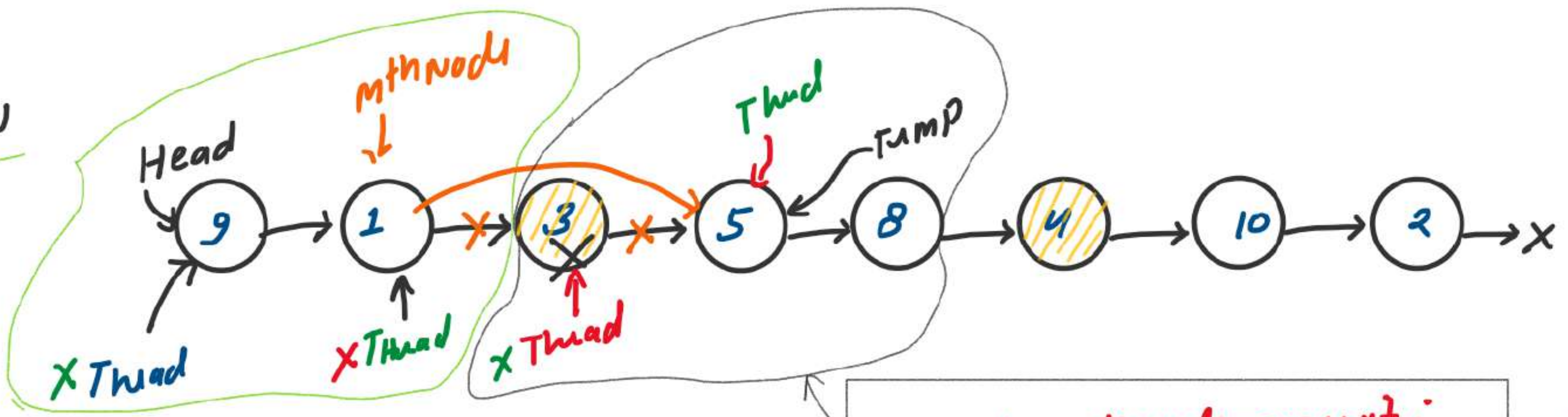
Ex 1





DRY RUN

M=2
N=1



Step:1

```
Node* Thead = Head;
for(int i=0; i<M-1; i++)
{
    Thead = Thead->next;
}
Node* Mth Node = Thead;
```

STEP 2

```
Thead = mthNode->next;
for(int i=0; i<N; i++)
{
    if(!Thead) Break; ***
    Node* Temp = Thead->next;
    delete Thead;
    Thead = Temp;
}
mthNode->next = Thead;
```

RUN TIME ERROR → Nth Node Available nahi Hai

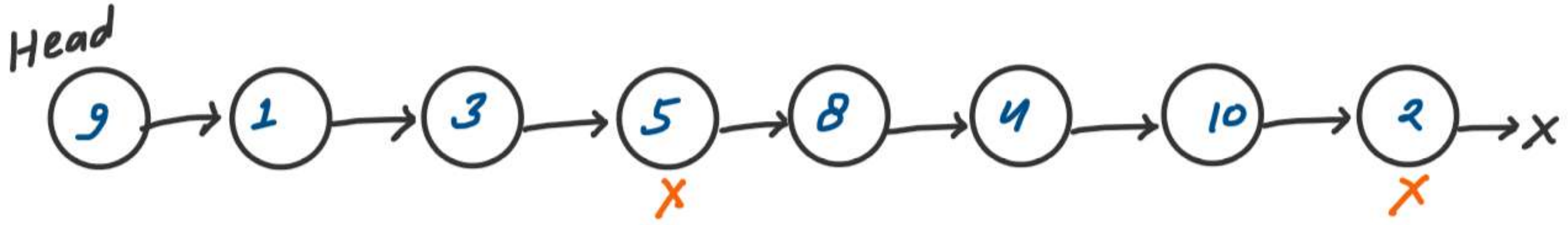
BASE Case

```
if(!head)
return
```

Step3

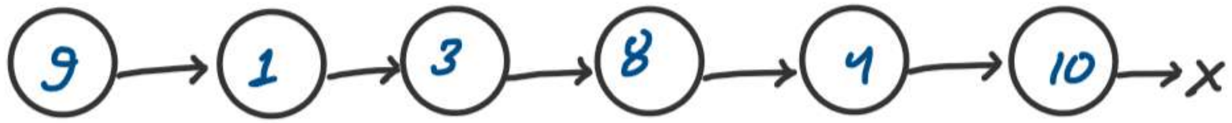
```
fun(Thead, M, N)
```
recursion call

Ex2



M=3 N=1

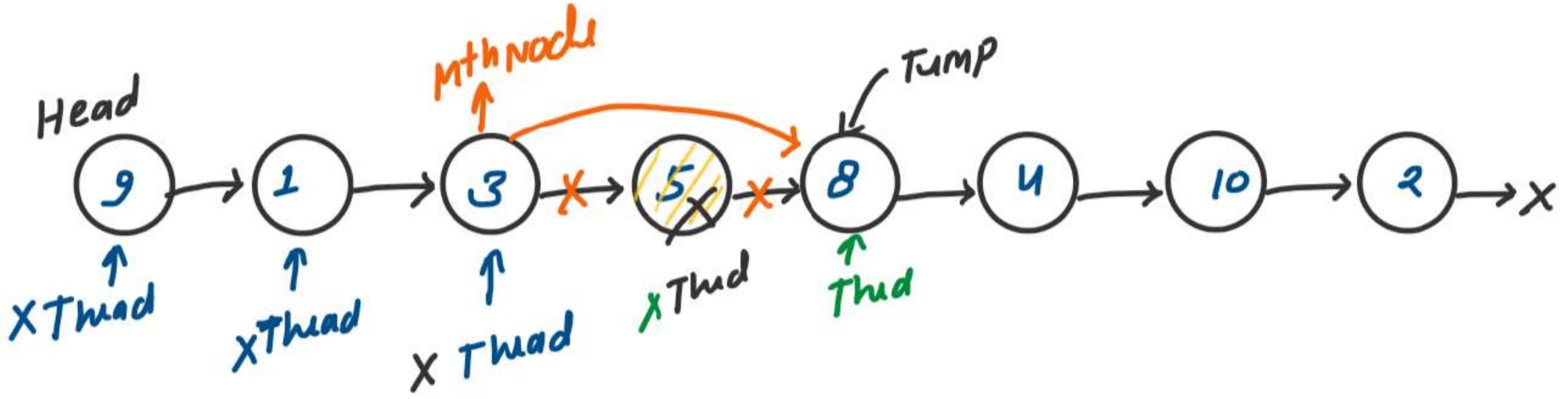
Output



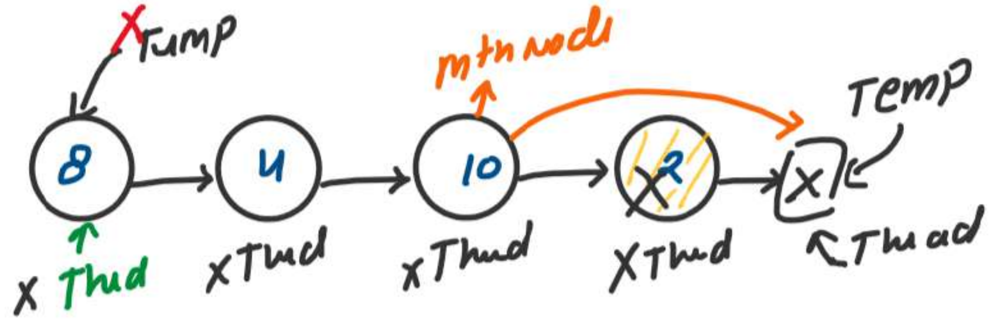
DRY RUN

M=3
N=1

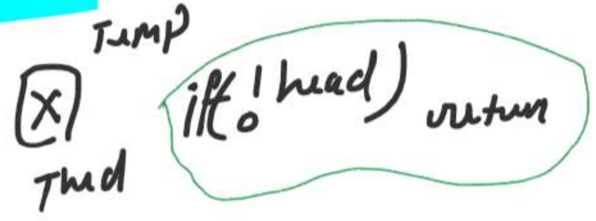
Iteration 1

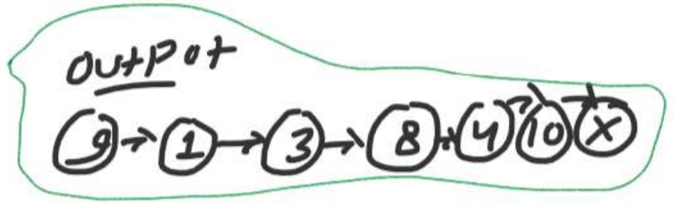


Iteration 2



Iter 3





Ex3

Head

$$\begin{tikzcd}
9 \arrow[r] & 1 \arrow[r] & 3 \arrow[r] & 5 \arrow[r] & 8 \arrow[r] & 4 \arrow[r] & 10 \arrow[r] & 2 \arrow[r] & X \\
 & & & & X & & & &
\end{tikzcd}$$

$M=4$ $N=1$

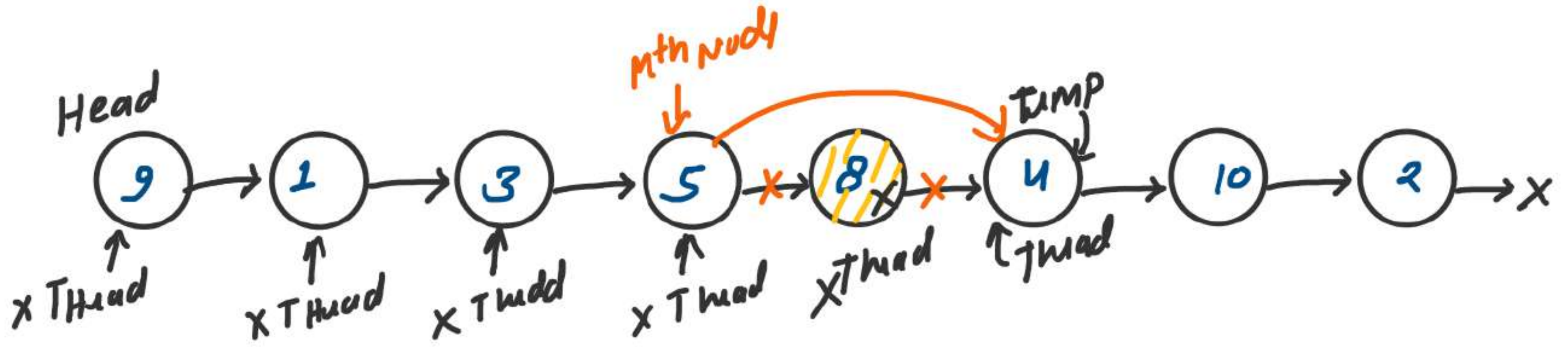
Output

$$\begin{tikzcd}
9 \arrow[r] & 1 \arrow[r] & 3 \arrow[r] & 5 \arrow[r] & 4 \arrow[r] & 10 \arrow[r] & 2 \arrow[r] & X
\end{tikzcd}$$

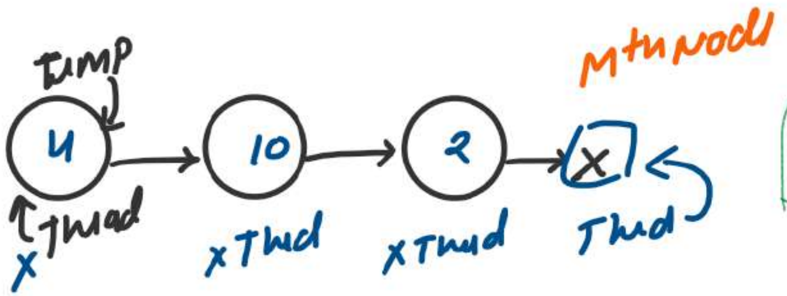
DRY RUN

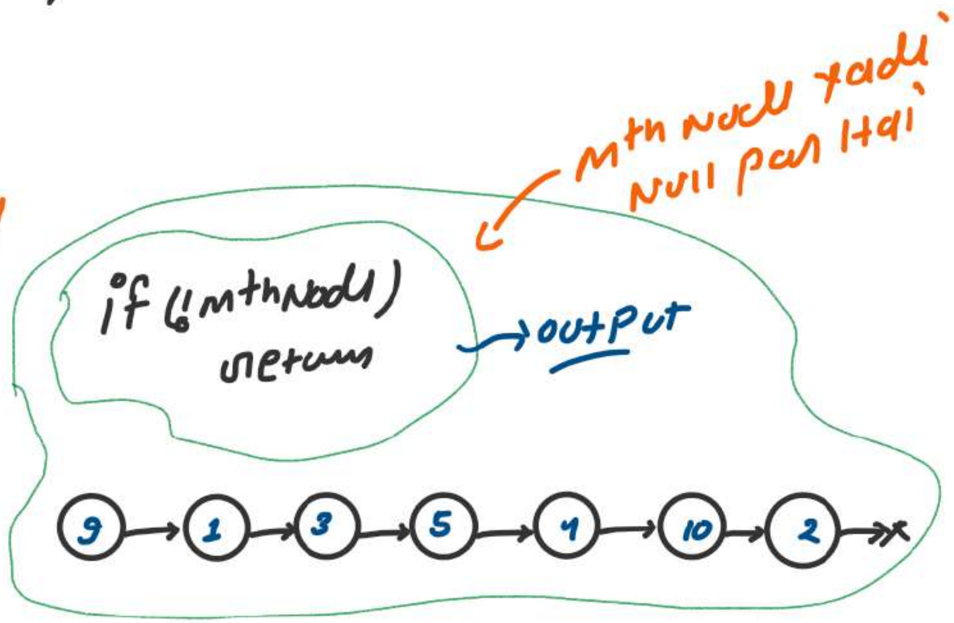
M = 4
N = 1

Iteration 1



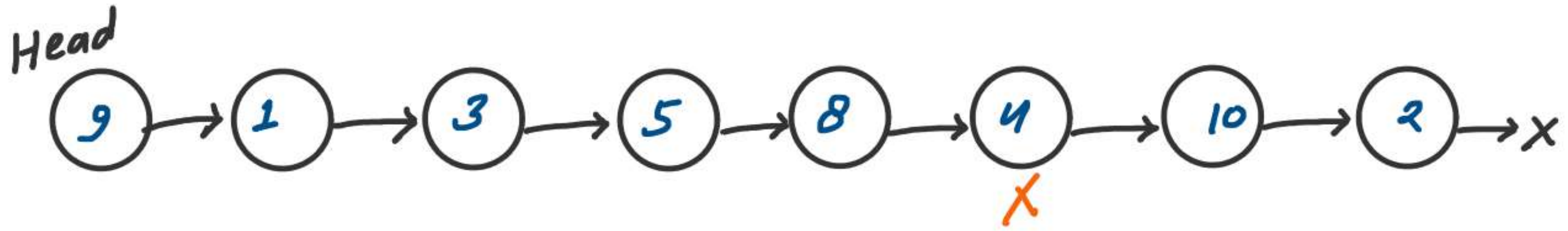
Iteration 2





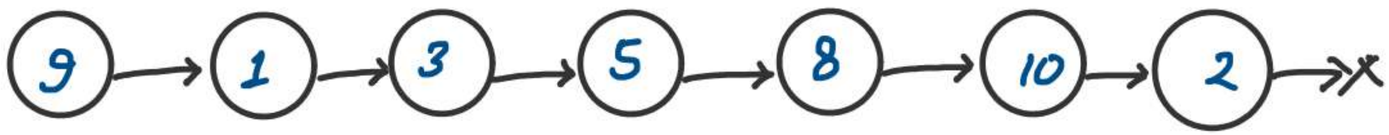
```
if (!mthNode)
    return
```

Ex 4



M=5 N=1

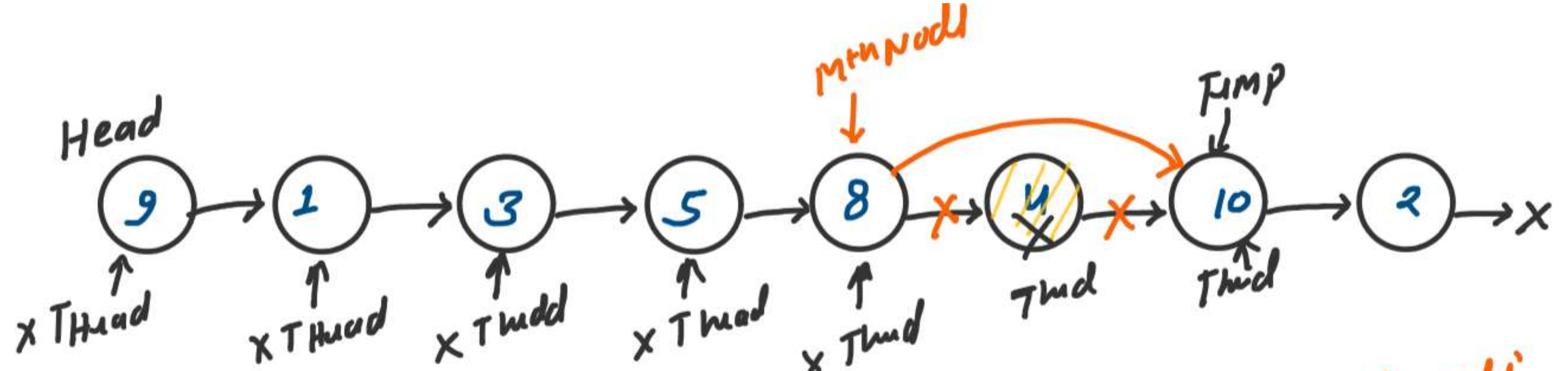
Output



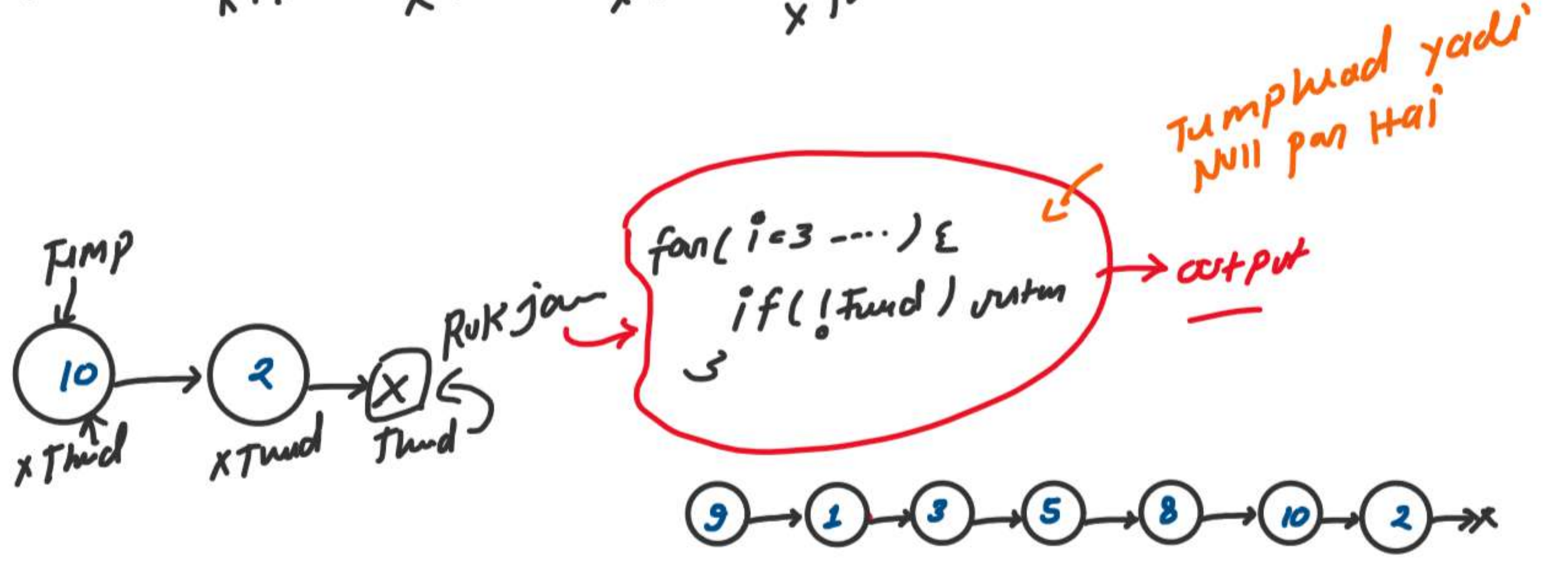
DRY RUN

$M=5$
$N=1$

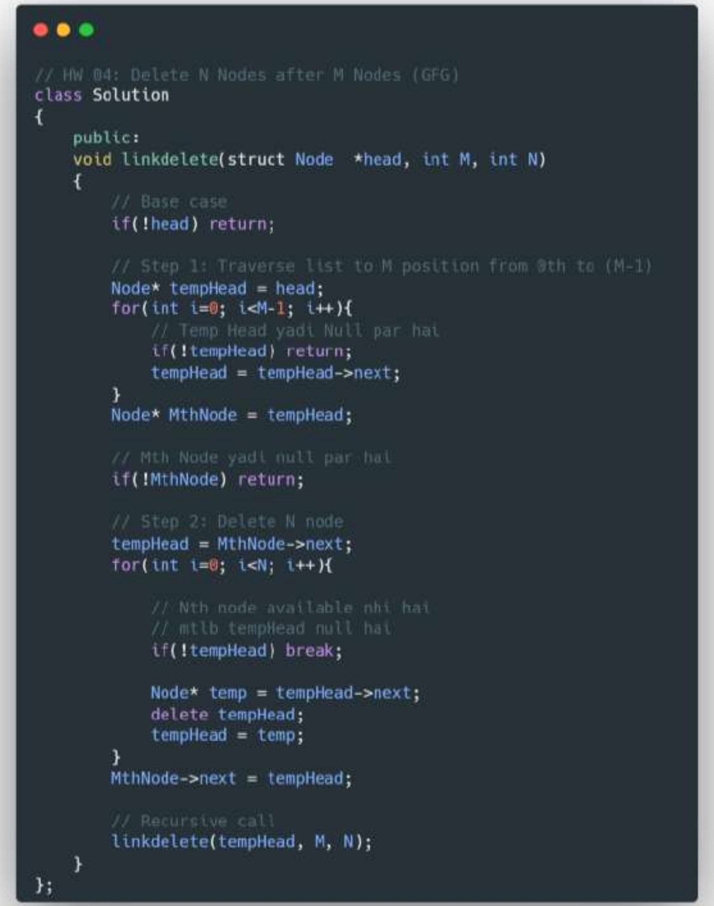
Iteration 1



Iteration 2



```
for( i=3 ....) {
    if(!Thead) return
}
```



```cpp
// HW 04: Delete N Nodes after M Nodes (GFG)
class Solution
{
    public:
    void linkdelete(struct Node  *head, int M, int N)
    {
        // Base case
        if(!head) return;

        // Step 1: Traverse list to M position from 0th to (M-1)
        Node* tempHead = head;
        for(int i=0; i<M-1; i++){
            // Temp Head yadi Null par hai
            if(!tempHead) return;
            tempHead = tempHead->next;
        }
        Node* MthNode = tempHead;

        // Mth Node yadi null par hai
        if(!MthNode) return;

        // Step 2: Delete N node
        tempHead = MthNode->next;
        for(int i=0; i<N; i++){

            // Nth node available nhi hai
            // mtlb tempHead null hai
            if(!tempHead) break;

            Node* temp = tempHead->next;
            delete tempHead;
            tempHead = temp;
        }
        MthNode->next = tempHead;

        // Recursive call
        linkdelete(tempHead, M, N);
    }
};
```

T.C. ⇒ O(N)
Where N is Numbers of nodes in the list.

S.C. ⇒ O(1)