
FACE
Prep

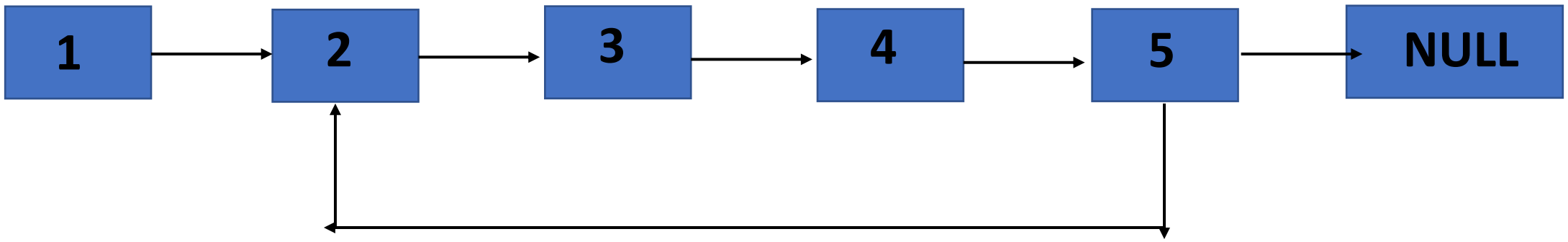
Loop Detection

Loop Detection or Detect Cycle in SLL

Problem: Given a linked list, check if the linked list has loop or not.

Below diagram shows a linear linked list(without loop).

Below diagram shows a linear linked list (with loop)



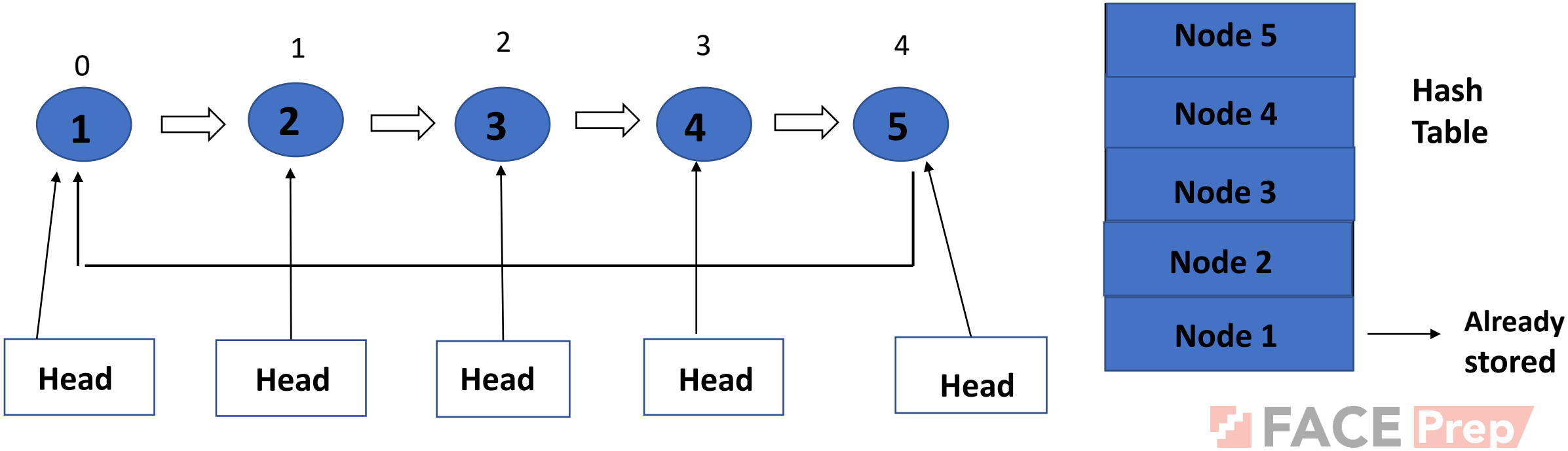
Solutions

1. Use Hashing

2. Floyd’s cycle finding(Using two pointer variables)

Use Hashing

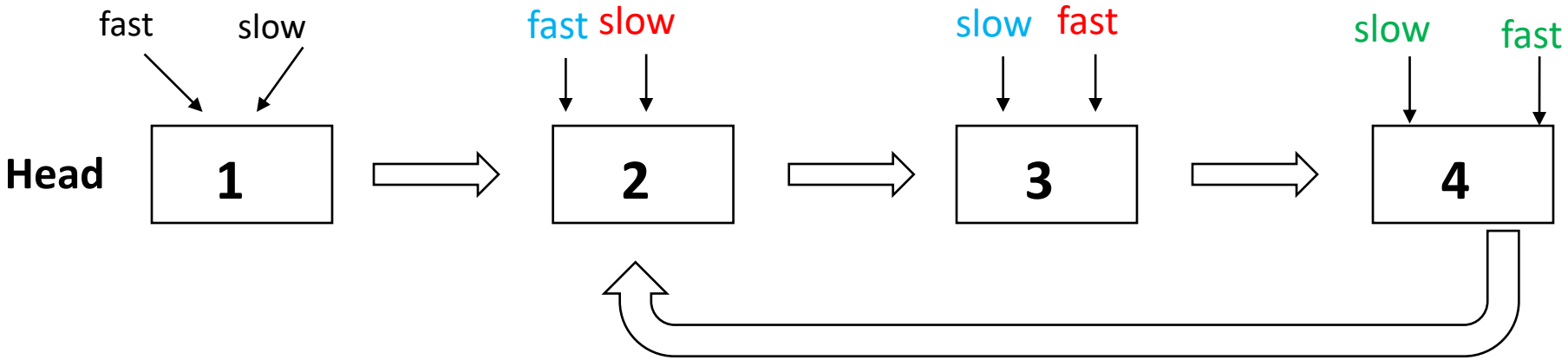
- Traverse the list one by one and keep putting the node addressed in a hash table.
- At any point, if NULL is reached then return false and if next of current node points to any of the previously stored nodes in Hashtable then return true.



Floyd’s cycle finding(Using two pointer variables)

- This algorithm used to find the loop in the given list.
- We will take two pointers, namely fast and slow. Fast pointer takes 2 steps ahead and slow pointer takes 1 step ahead.
- Iterate through the list until the fast pointer is equal to NULL. If the fast pointer is NULL, then there is no cycle present in the given list.
- During iteration, if fast pointer and slow pointer pointing same node then cycle is present in the given list.

Using two pointer variables



Slow and fast pointers starts from the first node.

slow ! = fast, then continue the process

slow pointer takes 1 step ahead and fast pointer taker 2 steps ahead.

slow !=fast

slow==fast, then the loop is detected

```
1   import java.util.*;
2   class Node {
3         int num;
4         Node next;
5         Node(int val) {
6               num=val;
7               next=NULL;
8         }
9   }
10  class Main {
11     static Node insertNode(Node head, int val) {
12        Node newNode = new Node(val);
13        if(head==null) {
14              head=newNode;
15              return head;
16        }
17        Node temp=head;
18        while(temp.next!=null)
19              temp=temp.next;
20        temp.next=newNode;
21        return head;
22     }
```

```
1       static void display(Node head)  {
2           Node temp = head;
3
4           while(temp.next!=null)  {
5               System.out.print(temp.num+"->");
6               temp=temp.next;}
7           System.out.println(temp.num+"->"+"NULL");
8       }
9
10      static void createCycle(Node head,int a,int b) {
11          int cnta = 0,cntb = 0;
12          Node p1 = head;
13          Node p2 = head;
14
15          while(cnta != a || cntb != b) {
16              if(cnta != a)
17               p1 = p1.next; ++cnta;
18
19              if(cntb != b)
20               p2 = p2.next; ++cntb; }
21           p2.next = p1; }
22
```

```
1       static boolean cycleDetect(Node head) {
2           if(head == null) return false;
3
4           Node fast = head;
5           Node slow = head;
6           while(fast.next != null && fast.next.next != null) {
7
8               fast = fast.next.next;
9               slow = slow.next;
10              if(fast == slow) return true;
11
12          }
13          return false;
14      }
15
16      public static void main(String args[]) {
17          Scanner sc = new Scanner(System.in);
18          int n=sc.nextInt();
19
20          Node head = null;
21
22
```

```
        for(int i=0;i<n;i++){
            int m=sc.nextInt();
            head=insertNode(head,m);
        }
        display(head);
        int a=sc.nextInt();
        createCycle(head,1,a);//creating cycle in the list
        if(cycleDetect(head) == true)
                System.out.println("Cycle detected");
        else
                System.out.println("Cycle not detected");
    }
}
```

THANK YOU

FACE Prep