链表是否有环（快慢指针）

```java
public class LinkListUtli {
        public static boolean hasCircle(LNode L)
        {
                if(L==null) return false;//单链表为空时，单链表没有环
                if(L.next==null) return false;//单链表中只有头结点，而且头结点的next为空，
                LNode p=L.next;//p表示从头结点开始每次往后走一步的指针
                LNode q=L.next.next;//q表示从头结点开始每次往后走两步的指针
                while(q!=null) //q不为空执行while循环
                {
                        if(p==q) return true;//p与q相等，单链表有环
                        p=p.next;
                        q=q.next.next;
                }
                return false;
        }
}
```

链表是否有环（找出环点）

拓展：判断带头结点的单链表是否有环，并找出环的入口结点

解题思路：

不妨设从头结点到环的入口结点需要走a步即环外包括头结点在内总共有a个结点，环中有b个结点，假设slow走了s步后，slow与fast第一次相遇。它们肯定是在环内相遇，而且相遇时slow没有从环的入口结点再次走到环的入口结点,假设在环中距离环入口结点d步长距离相遇，设相遇结点为meet。

此时令头结点指向fast，让slow与fast每次往后走一步，当它们再次第一次相遇时，相遇的结点就是环的入口结点。

证明如下：

当slow再走s步后会再次到meet结点，而此时fast走了s步后也会首次到达meet结点，它们相遇，因为两者都是每次同时走一步，那么从fast进入环中开始，fast与slow就一直相遇，它们首次相遇的结点就是环的入口结点。

```java
public class LinkListUtli {
        //当单链表中没有环时返回null，有环时返回环的入口结点
        public static LNode searchEntranceNode(LNode L)
        {
                if(L==null) return null;//单链表为空时，单链表没有环
                if(L.next==null) return null;//单链表中只有头结点，而且头结点的next为空，
                LNode p=L.next;//p表示从头结点开始每次往后走一步的指针
                LNode q=L.next.next;//q表示从头结点开始每次往后走两步的指针
                while(q!=null) //q不为空执行while循环
                {
                        if(p==q) break;//p与q相等，单链表有环
                        p=p.next;
                        q=q.next.next;
                }
                if(q==null) return null;

                //这里之所以没有向上面一样，先让p,q走一步再进入循环判断，是因为头结点可能就
                q=L;
                while(q!=null)
                {
                        if(p==q) return p;//返回环中入口结点
                        p=p.next;
                        q=q.next;
                }
                return null;
        }
}
```