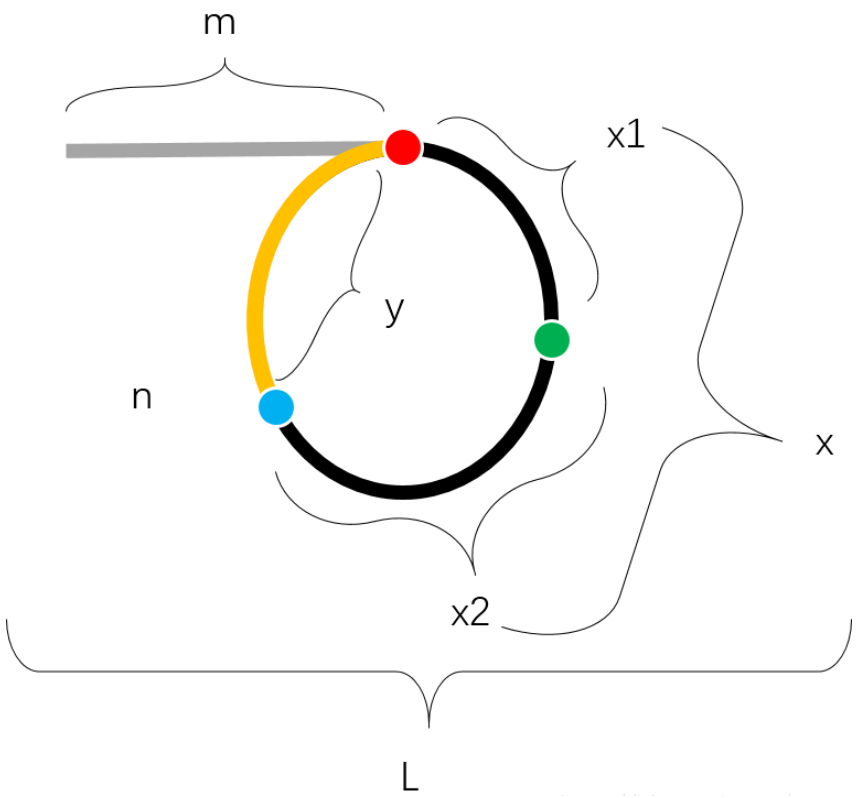
差速法:

```
1.  ListNode* EntryNodeOfLoop(ListNode* pHead) {
2.      if(pHead == null|| pHead.next == null|| pHead.next.next == null)
3.          return null;
4.      ListNode* fast = pHead->next->next;
5.      ListNode* slow = pHead->next;
6.      while(fast! = slow) {
7.          if(fast->next != null && fast->next->next != null) {
8.              fast = fast->next->next;
9.              slow = slow->next;
10.         } else {
11.             return null;
12.         }
13.     }
14.     fast = pHead;
15.     while(fast != slow) {
16.         fast = fast->next;
17.         slow = slow->next;
18.     }
19.     return slow;
20. }
```

证明:



上图是一个简单有环单链表的模型图，总长度为 L，无环部分长度（图中灰色部分）为 m，环的长度（图中黑色加橙色部分）为 n，红点为环的入口节点。

现在有两个指针，一个每次移动一个节点，称为慢指针，另一个的速度是前者的两倍，称为快指针。两个指针同时从链表头节点出发，它们必在环中相遇。图中绿色节点为慢指针刚到达环入口节点时快指针的位置，此时距离慢指针的距离（图中红绿两点间黑色部分）为 x1，蓝色点表示快慢指针在环中相遇的位置，此时距离环的入口节点距离（图中红蓝两点间黑色部分）为 x，沿着运动方向距离入口节点距离（图中橙色部分）还有 y。

上面是对图的说明，下面开始证明一个结论：

$$m = k * n + y. \text{ (k 为正整数)}$$

当快慢指针相遇时，假设慢指针走了 t 步，那么快指针一定走了 2t 步。

当慢指针进入以后，快指针一定会在慢指针走完一圈之前追上它。因为最坏的情况是慢指针进入环时快指针刚好落后它一整圈，这样慢指针走完一圈快指针刚好追上慢指针。

快慢指针相遇时，慢指针走过的距离是：

$$t = m + x$$

当慢指针刚进入环入口时，快指针走过的的距离是：

$$t1 = m + k * n + x1. \text{ (k 为正整数)}$$

当快慢指针相遇时，快指针又走过了：

$$t2 = n + x2$$

所以快慢指针相遇时，快指针一共走过了：

$$t1 + t2 = m + k * n + x1 + n + x2 = m + (k + 1) * n + x = m + k * n + x. \text{ (k 为正整数)}$$

而相遇时快指针走过的距离是慢指针的两倍。

$$2 * t = t1 + t2$$

$$2(m + x) = m + k * n + x$$

$$m = k * n - x$$

$$m = (k - 1) * n + (n - x)$$

$$m = k * n + y. \quad \text{(k 为正整数)}$$

这个结论说明，从头节点到环入口的距离等于快慢指针相遇处继续走到环入口（图中橙色部分）的距离加上环长度的整倍数。如果有一个人从链表头节点开始每次移动一个节点地往后走，另一个人从快慢指针相遇处（图中蓝色点）以同样的速度往前走，结果就是，两人相遇在环的入口节点处。