1

```
bool isCircle(Listnode *head){
        Listnode *slow=head,*fast=head;
        while(fast){
                fast=fast->next;
                if(fast)
                fast=fast->next;
                slow=slow->next;
                if(slow==fast)
                        break;
        }
        return fast==nullptr?false:true;
}
```

空间复杂度:
O(1)
时间复杂度
假如没有环时间复杂度为 $O(n)$;假如有环,令 K 为从 head 开始到环开始地方的长度,L 为环的长度,令 $D = K \mod L$,时间复杂度为 $O(K+(L-D))$。
总结来说，时间复杂度 $O(n)$

2

2.1

设计:
跳表共有 n 层单链表组成。
level1 层单链表包含了原单链表的所有元素
level2 层单链表包含了 $\frac{n}{2}$ 元素，相邻元素的间隔为 2
level3 层单链表包含了 $\frac{n}{4}$ 元素，相邻元素间隔为 4
以此类推
n 的大小由原单链表的长度 L 决定
第二步
将每层的链表连起来
具体操作:

```
struct Skiplistnode{
        int val;
        Skiplistnode *next,*down;
}
```

其中 next 指针指向同 level 链表的下一个元素，down 指针指向 level-1 层链表的相同元素。（这个由构造保证一定存在）

这里，我们首先需要将原链表进行排序。查找第 i 个元素的伪代码

```
fuction search(Skiplistnode *cur,int tar){
    if cur->val == tar
        return
        if cur->next == nullptr || cur->next->val<=tar
           search(cur->next,tar)
        search(cur->down,tar)
}
```

2.2

时间复杂度分析：

由于 SkipList 的构造可知，通过判断 cur->next->val 与 tar 的大小，将整个 search 的范围缩小了一半

可以得到递推方程

T(n)=T(n/2)+O(1)

得到 T(n)=O(logn)