2022.9.18

数算作业二

1.

Input: Node* head.

Output: void

如果cur→next与cur的值相同，就删除cur→next

```
Node* cur=head.
while(cur!=NULL)
    while(cur→next→val==cur→val)
        delete(cur→next);
    cur=cur→next;
```

其中变量只使用cur，空间复杂度为O(1)

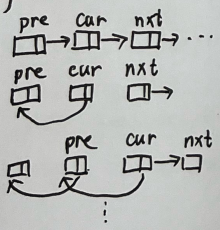
delete操作与cur赋值操作均最多进行n次

从而 $f(n)<2n$

于是 $f(n)=O(n)$ 为时间复杂度

2.

```
Input: Node* head.
Output: Node* head_reverse
{
Node* pre=head;
Node* cur=head→next;
Node* nxt=head→next→next;
head→next=NULL;
cur→next=pre;
while(nxt!=NULL) //若nxt不为空则将所有3个指针后移并更换cur→next
    pre=cur;
    cur=nxt;
    nxt=nxt→next;
    cur→next=pre;
return cur;
}
```




只使用3个Node*类型指针，空间复杂度为O(1)

整个循环相当于cur指针遍历一次链表，每次操作有固定的赋值次数，时间复杂度为O(n)

3.

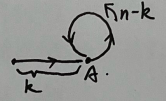
考虑利用快慢指针

如果快指针与慢指针指向同一个节点，则有环

```
Input: Node* head
Output: bool has_ring
{
Node* fast_p=head;
Node* slow_p=head;
while(fast_p!=NULL && slow_p!=NULL)
    slow_p=slow_p→next;
    fast_p=fast_p→next→next;
    if(slow_p==fast_p)
        return true;
return false;
}
```

只使用2个指针，空间复杂度为O(1)

对于时间复杂度考虑，如下有环链表，总长为n.




在慢指针进入环时，快指针在A后k处.

再经过 $(n-k)-k\%(n-k)$ 次循环快指针追上慢指针

因此总次数 $f(n)=n-k\%(n-k)$

$<n$

从而时间复杂度为O(n)

1.

```
Input: a[N]
int pos=0;
for(i from 1 to N)
    if(a[pos]!=a[i])
        a[1+pos]=a[i];
```

只使用变量pos与i，空间复杂度为O(1)

循环经过1~N，时间复杂度为O(n)