/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     struct ListNode \*next;

 \* };

 \*/

void reorderList(struct ListNode\* head){

struct ListNode\* index;

struct ListNode\* newHead;

struct ListNode\* tmp;

struct ListNode\* pre\_tmp;

index=head;

tmp=head;

newHead=index;

int count=0;

int mid=0;

int mid\_val=-1;

while( index != NULL ){//取出linked list中間之前的nodes

        mid=count/2;

        //printf("count=%d mid=%d \n",count,mid);

        if( mid > mid\_val){

                mid\_val=mid;

                //printf("mid=%d %p %d \n",mid,tmp,tmp->val);

                pre\_tmp=tmp;//pre\_tmp會記錄中間的上一個node

                tmp=tmp->next;//tmp會記錄中間的node

        }

        index=index->next;

        count++;

}

if( mid == 0){

        return pre\_tmp;

}

//printf("\n");

//printf("%p %d mid=%d\n",pre\_tmp,pre\_tmp->val,mid);

pre\_tmp->next=NULL;//pre\_tmp會記錄中間的上一個node之後斷開

//printf("%p %d mid=%d\n",tmp,tmp->val,mid);

count=0;

struct ListNode\* tmp\_invert=NULL;

struct ListNode\* tmp2=NULL;

//printf("\n");

while( tmp != NULL ){//中間之後的nodes做reverse

        //printf("%p %d \n",tmp,tmp->val);

        if(tmp\_invert == NULL){

                tmp\_invert=tmp;

                tmp=tmp->next;

                tmp\_invert->next=NULL;

        }else{

            tmp2=tmp\_invert;

            tmp\_invert=tmp;

            tmp=tmp->next;

            tmp\_invert->next=tmp2;

        }

}

//printf("\n");

// while( tmp\_invert != NULL ){

//         printf("%p %d \n",tmp\_invert,tmp\_invert->val);

//         tmp\_invert=tmp\_invert->next;

// }

// printf("\n");

//以下要做merge的動作，將revese(tmp\_invert)跟中間之前的node(newHead)做merge

struct ListNode\* res\_index=NULL;//res\_index是merge用的指標

res\_index=newHead;//先丟要merge的第一個node(這題先丟newHead)

newHead=newHead->next;

struct ListNode\* res=res\_index;//res是之後要給答案的開頭address

//printf("res %p \n",res);

//printf("%p %d \n",res\_index,res\_index->val);

//printf("\n");

while(1){

        //printf("%p %d \n",tmp\_invert,tmp\_invert->val);

        if( tmp\_invert !=NULL){

                res\_index->next=tmp\_invert;

                tmp\_invert=tmp\_invert->next;

                res\_index=res\_index->next;

                //printf("%p %d \n",res\_index,res\_index->val);

        }

        if( newHead !=NULL){

                res\_index->next=newHead;

                res\_index=res\_index->next;

                newHead=newHead->next;

                //printf("%p %d \n",res\_index,res\_index->val);

        }

        if( count == mid){//當數到中間位數mid時結束迴圈

                //printf("break\n");

                res\_index->next=NULL;

                break;

        }

        count++;

}

//printf("\n");

// while( res != NULL ){

//         //printf("res %p %d \n",res,res->val);

//         res=res->next;

// }

//printf("\n");

return res;

}