```
/*
无输入
输出:
1 3
0 2 4 6 8
8 6 4 3 2 1 0
*/
#include <iostream>
using namespace std;
const int MinNumber=-1000000;
struct Node
{
        Node(int x)
        {
                value=x;
                next=NULL;
        }
        int value;
        Node* next;
};
Node*  Merge(Node* a,Node *b)
{
        Node *temp;
        Node *p=a,*q=b;
        Node *head = a->value >= b->value?b:a; // 选择a b中最小的值作为头节点
        while(p!=NULL&&q!=NULL) // a b 只要有一个遍历完成，就不用继续选择
        {
                if(p->value < q->value) //p比q小，由q指向p
                {
                        temp=p->next;
                        p->next=q;
```

```
                p=temp;
            }
            else
            {
                temp=q->next;
                q->next=p;
                q=temp;
            }
        }
        return head; // 返回的head 是递增的
}
Node* Reverse(Node* first) // 倒置链表
{
        Node *p=first,*q=p->next;
        first->next=NULL; //记得把第一个的next赋值为NULL
        while(q!=NULL)
        {
            Node* QNextTemp=q->next;
            q->next=p;
            p=q;
            q=QNextTemp;
        }
        return p;
}
void outputNode(Node* head) // 输出链表
{
    Node *p=head;
    while(p!=NULL)
    {
        cout<<p->value<<" ";
        p=p->next;
```

```
    }
    cout<<endl;
}
int main()
{
        Node* a=new Node(1);
        Node *p=a;
        for(int i=3;i<=3;i+=2)
        {
                Node * t=new Node(i);
                p->next=t;
                p=t;
        }
        outputNode(a);
        Node* b=new Node(0);
        p=b;
        for(int i=2;i<=8;i+=2)
        {
                Node * t=new Node(i);
                p->next=t;
                p=t;
        }
        outputNode(b);
        Node * head=Merge(a,b);
        head=Reverse(head);
        outputNode(head);
        return 0;
        /**********分析**********
        设a的长度为m,b的长度为n,算法的复杂度为0(2*(m+n))
        **********/
}
```