双链表（数组实现）

实现一个双链表，双链表初始为空，支持 5 种操作：

1. 在最左侧插入一个数；
2. 在最右侧插入一个数；
3. 将第 k 个插入的数删除；
4. 在第 k 个插入的数左侧插入一个数；
5. 在第 k 个插入的数右侧插入一个数

现在要对该链表进行 M 次操作，进行完所有操作后，从左到右输出整个链表。

注意:题目中第 k 个插入的数并不是指当前链表的第 k 个数。例如操作过程中一共插入了 n 个数，则按照插入的时间顺序，这 n 个数依次为：第 1 个插入的数，第 2 个插入的数，…第 n 个插入的数。

输入格式

第一行包含整数 M，表示操作次数。

接下来 M 行，每行包含一个操作命令，操作命令可能为以下几种：

1. `L x`，表示在链表的最左端插入数 x。
2. `R x`，表示在链表的最右端插入数 x。
3. `D k`，表示将第 k 个插入的数删除。
4. `IL k x`，表示在第 k 个插入的数左侧插入一个数。
5. `IR k x`，表示在第 k 个插入的数右侧插入一个数。

输出格式

共一行，将整个链表从左到右输出。

数据范围

$1 \leq M \leq 100000$
所有操作保证合法。

输入样例：

```
10
R 7
D 1
L 3
IL 2 10
D 3
IL 2 7
L 8
R 9
IL 4 7
IR 2 2
```

输出样例：

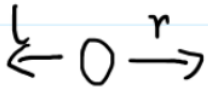
```
8 7 7 3 2 9
```

代码：

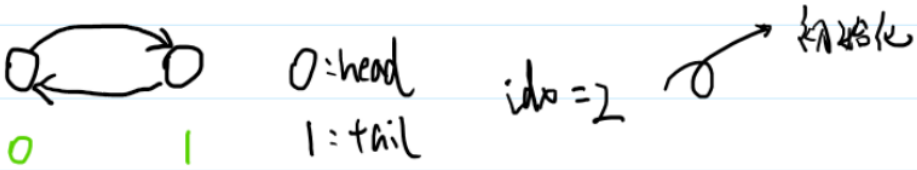
```
#include <iostream>

using namespace std;

const int N=100010;

int e[N],l[N],r[N],idx=0;
```

```
void insertL(int x)
{
    l[idx]=0;//新节点->pre=head
    r[idx]=r[0];//新节点->next=head->next
    r[0]=idx;//head->next=新节点
    l[r[idx]]=idx;//新节点->next->pre=新节点
    e[idx++]=x;//add x
}

void insertR(int x)
{
    r[idx]=1;//新节点->next=tail
    l[idx]=l[1];//新节点->pre=tail->pre
    r[l[idx]]=idx;//新节点->pre->next=新节点
    l[1]=idx;//tail->pre=新节点
    e[idx++]=x;//add x
}

void deleteK(int k)
{
    r[l[k]]=r[k];//k->pre->next=k->next
    l[r[k]]=l[k];//k->next->pre=k->pre
}

void insertKL(int k,int x)
{
    l[idx]=l[k];//新节点->pre=k->pre
    r[idx]=k;//新节点->next=k
    r[l[idx]]=idx;//新节点->pre->next=新节点
    l[k]=idx;//k->pre=新节点
    e[idx++]=x;//add x
}

void insertKR(int k,int x)
{
    r[idx]=r[k];//新节点->next=k->next
    l[idx]=k;//新节点->pre=k
    l[r[idx]]=idx;//新节点->next->pre=新节点
    r[k]=idx;//k->next=新节点
    e[idx++]=x;//add x
}

void init()
{
    r[0]=1;//head->next=tail
    l[1]=0;//tail->pre=head
    idx=2;//add two nodes
}

int main()
{
    ios::sync_with_stdio(false);
    init();
    int m;
    cin>>m;

    while(m-->0)
```

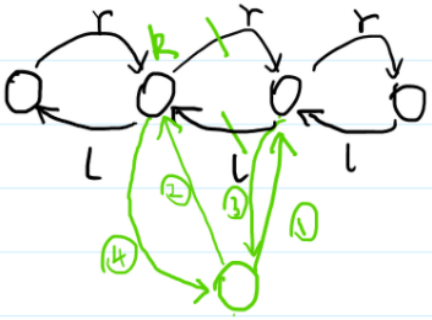
```
    {

        string command;
        int k,x;
        cin>>command;
        if(command=="L")
        {
            cin>>x;
            insertL(x);
        }
        else if(command=="R")
        {
            cin>>x;
            insertR(x);
        }
        else if (command=="D")
        {
            cin>>k;
            deleteK(k+1);//因为初始化加了两个节点，所以第k个数的下标为k+2-1
        }
        else if (command=="IL")
        {
            cin>>k>>x;
            insertKL(k+1,x);
        }
        else if(command=="IR")
        {
            cin>>k>>x;
            insertKR(k+1,x);
        }


    }
    for(int i=r[0];i!=1;i=r[i]) cout<<e[i]<<' ';//从头开始,当i!=tail时输出
    cout<<endl;
    return 0;

}
```



e[N] , l[N] , r[N] , idx



在k右边插入 add(k, x)

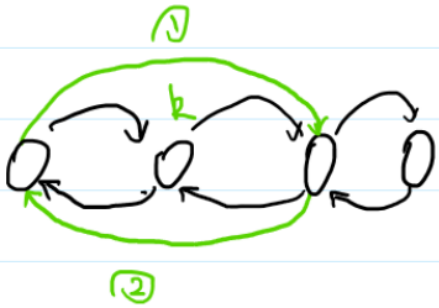


e[idx] = x
① r[idx] = r[k]
② l[idx] = k
③ l[r[k]] = idx
④ r[k] = idx

插入

在k左边插入 → add(l[k], x)

删除第k个点 remove(k)



r[l[k]] = r[k]
l[r[k]] = l[k]

```
#include<iostream>

using namespace std;

const int N = 1e5 + 10;

int m;
int e[N], l[N], r[N];
```

```
int idx;


//! 初始化
void init()
{
    l[1] = 0, r[0] = 1;//* 初始化 第一个点的右边是 1   第二个点的左边是 0
    idx = 2;//! idx 此时已经用掉两个点了
}

//* 在第 k 个点右边插入一个 x
void add(int k, int x)
{
    e[idx] = x;
    l[idx] = k;
    r[idx] = r[k]; //todo 这边的 k 不加 1 ， 输入的时候 k+1 就好
    l[r[k]] = idx;
    r[k] = idx;
    idx++;
}//! 当然在 k 的左边插入一个数 可以再写一个 ， 也可以直接调用我们这个函数，在 k 的左边插入一个数 等价于在 l[k] 的右边插入一个数 add(l[k],x)

//*删除第 k个 点
void remove(int k)
{
    r[l[k]] = r[k];
    l[r[k]] = l[k];
}

int main(void)
{
    ios::sync_with_stdio(false);
    cin >> m;

    init();

    while(m--)
    {
        string op;
        cin >> op;
        int k, x;
        if(op=="R")
        {
            cin >> x;
            add(l[1], x); //!   0和 1 只是代表 头和尾  所以   最右边插入 只要在  指向 1 的 那个点的右边插入就可以了
        }
        else if(op=="L")//! 同理  最左边插入就是 在指向 0的数的左边插入就可以了   也就是可以直接在 0的 有右边插入
        {
            cin >> x;
            add(0, x);
        }
        else if(op=="D")
        {
            cin >> k;
            remove(k + 1);
        }
```

```
        else if(op=="IL")
        {
            cin >> k >> x;
            add(l[k + 1], x);
        }
        else
        {
            cin >> k >> x;
            add(k + 1, x);
        }
    }
    for(int i = r[0]; i != 1; i = r[i]) cout << e[i] << ' ';

    return 0;
}
```


y总代码：

```
#include <iostream>

using namespace std;

const int N = 100010;

int m;
int e[N], l[N], r[N], idx;

// 在节点a的右边插入一个数x
void insert(int a, int x)
{
    e[idx] = x;
    l[idx] = a, r[idx] = r[a];
    l[r[a]] = idx, r[a] = idx ++ ;
}

// 删除节点a
void remove(int a)
{
    l[r[a]] = l[a];
    r[l[a]] = r[a];
}

int main()
{
    cin >> m;

    // 0是左端点，1是右端点
    r[0] = 1, l[1] = 0;
    idx = 2;

    while (m -- )
    {
```

```
        string op;
        cin >> op;
        int k, x;
        if (op == "L")
        {
            cin >> x;
            insert(0, x);
        }
        else if (op == "R")
        {
            cin >> x;
            insert(l[1], x);
        }
        else if (op == "D")
        {
            cin >> k;
            remove(k + 1);
        }
        else if (op == "IL")
        {
            cin >> k >> x;
            insert(l[k + 1], x);
        }
        else
        {
            cin >> k >> x;
            insert(k + 1, x);
        }
    }

    for (int i = r[0]; i != 1; i = r[i]) cout << e[i] << ' ';
    cout << endl;

    return 0;
}
```