双向链表

```
class ListsNode
{
public:
    int val;
    ListsNode* next;
    ListsNode* pre;
    ListsNode(int v, ListsNode* n, ListsNode* p):val(v),next(n),pre(p){}
};


class MyLinkedList {
public:
    ListsNode* root;
    ListsNode* trail;
    int size;

    /** Initialize your data structure here. */
    MyLinkedList() {
        root=nullptr;
        trail=nullptr;
        size=0;
    }

    /** Get the value of the index-th node in the linked list. If the index is
invalid, return -1. */
    int get(int index) {
        int temp=0;
        ListsNode* cur=root;
        while(cur!=nullptr)
        {
            if(temp==index)
            {
                return cur->val;
            }
            cur=cur->next;
            temp++;
        }
        return -1;
    }

    /** Add a node of value val before the first element of the linked list.
After the insertion, the new node will be the first node of the linked list. */
    void addAtHead(int val) {
        if(root!=nullptr)
        {
            ListsNode* newNode=new ListsNode(val, root, nullptr);
            root->pre=newNode;
            root=newNode;
        }
        else
        {
            root=new ListsNode(val, nullptr,nullptr);
            trail=root;
        }
        size++;
    }

    /** Append a node of value val to the last element of the linked list. */
    void addAtTail(int val) {
        if(trail!=nullptr)
        {
            ListsNode* newNode = new ListsNode(val, nullptr, trail);
            trail->next=newNode;
            trail=newNode;
        }
        else
        {
            trail=new ListsNode(val, nullptr, nullptr);
            root=trail;
        }
        size++;
    }

    /** Add a node of value val before the index-th node in the linked list. If
index equals to the length of linked list, the node will be appended to the end
of linked list. If index is greater than the length, the node will not be
inserted. */
    void addAtIndex(int index, int val) {
        if(index<=0)
        {
            addAtHead(val);
            return;
        }
        if(index==size)
        {
            addAtTail(val);
            return;
        }

        int temp=0;
        ListsNode* pre=nullptr;
        ListsNode* cur=root;
        while(cur!=nullptr)
        {
            if(temp==index)
            {
                ListsNode* newNode=new ListsNode(val, cur, pre);
                if(pre!=nullptr)
                {
                    pre->next=newNode;
                }
                cur->pre=newNode;
                size++;
                return;
            }
            pre=cur;
            cur=cur->next;
            temp++;
        }
    }
    /** Delete the index-th node in the linked list, if the index is valid. */
    void deleteAtIndex(int index) {
        int temp=0;
        ListsNode* pre=nullptr;
        ListsNode* cur=root;
        if(index==0)
        {
            ListsNode* old=root;
            root=root->next;
            if(root!=nullptr)
            {
                root->pre=nullptr;
            }
            delete old;
            size--;
            return;
        }
        if(index==size-1)
        {
            ListsNode* old=trail;
            trail=trail->pre;
            if(trail!=nullptr)
            {
                trail->next=nullptr;
            }
            delete old;
            size--;
            return ;
        }
        while(cur!=nullptr)
        {
            if(temp==index)
            {
                ListsNode* old=cur;
                if(pre!=nullptr)
                {
                    pre->next=cur->next;
                }
                if(cur->next!=nullptr)
                {
                    cur->next->pre=pre;
                }
                delete old;
                size--;
                return;
            }
            pre=cur;
            cur=cur->next;
            temp++;
        }
    }
};
/**
 * Your MyLinkedList object will be instantiated and called as such:
 * MyLinkedList* obj = new MyLinkedList();
 * int param_1 = obj->get(index);
 * obj->addAtHead(val);
 * obj->addAtTail(val);
 * obj->addAtIndex(index,val);
 * obj->deleteAtIndex(index);
 */
```