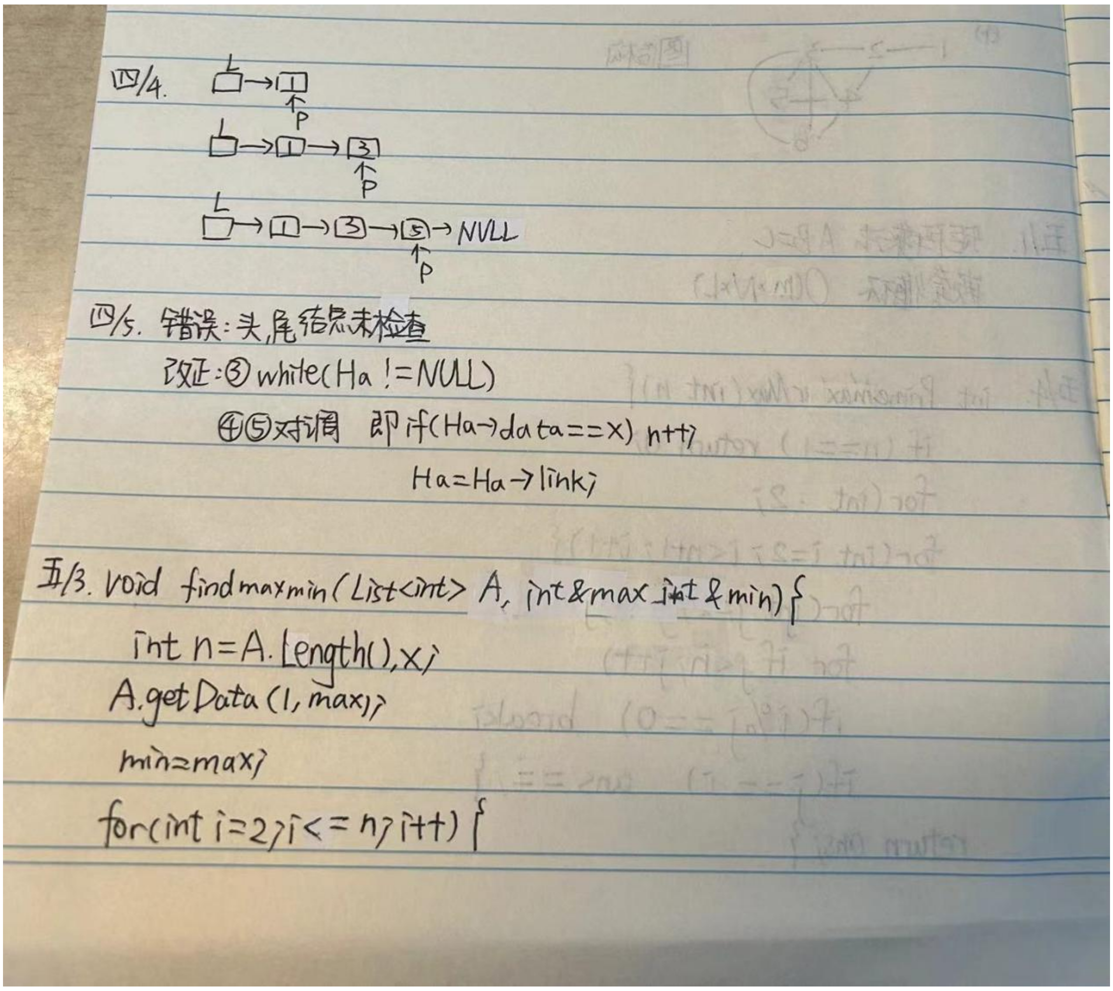


四/4.

```
L
□→[1]
   ↑p
L
□→[1]→[3]
        ↑p
L
□→[1]→[3]→[5]→NULL
            ↑p
```

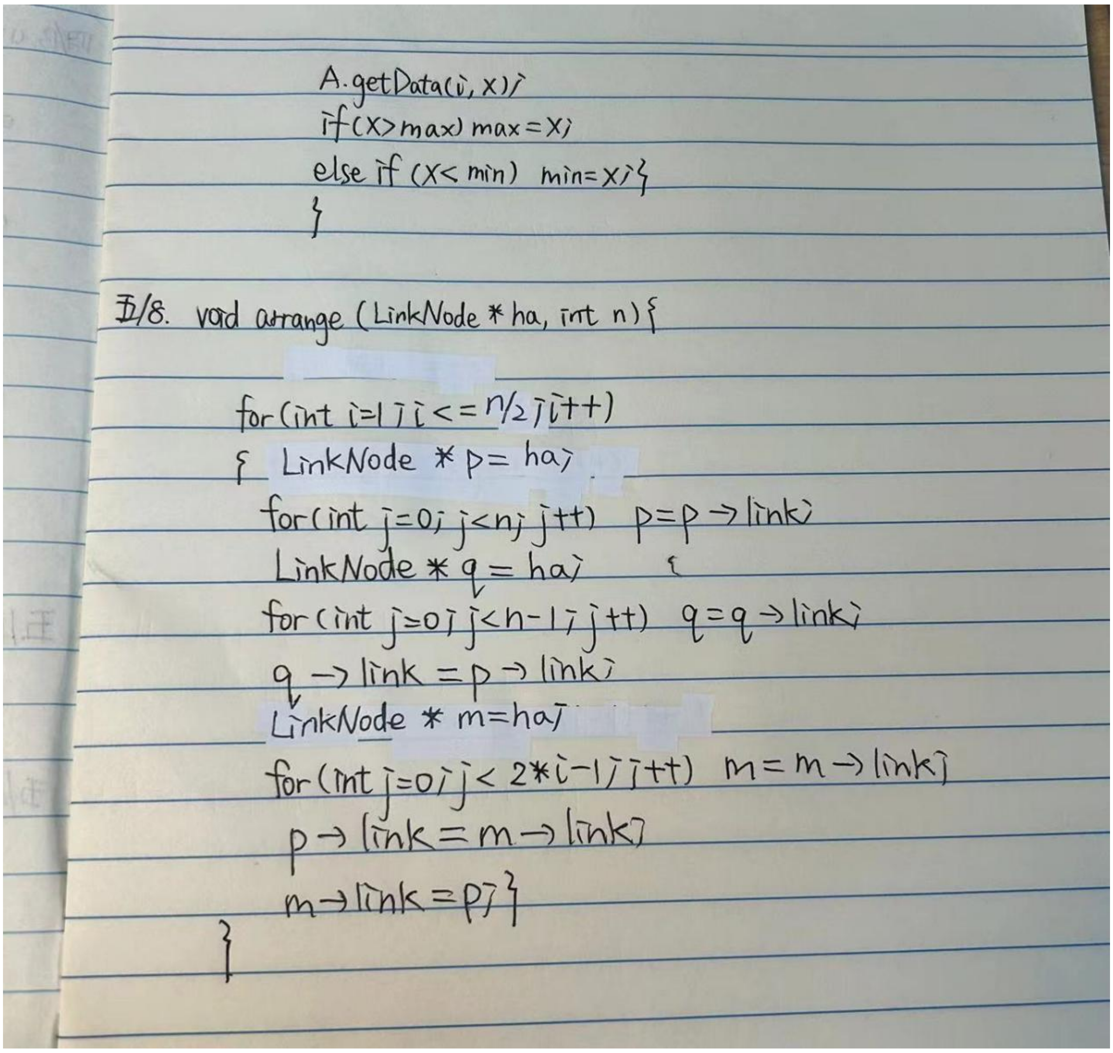
四/5. 错误：头,尾结点未检查

改正：③ while(Ha !=NULL)

④⑤对调 即 if(Ha→data==x) n++;

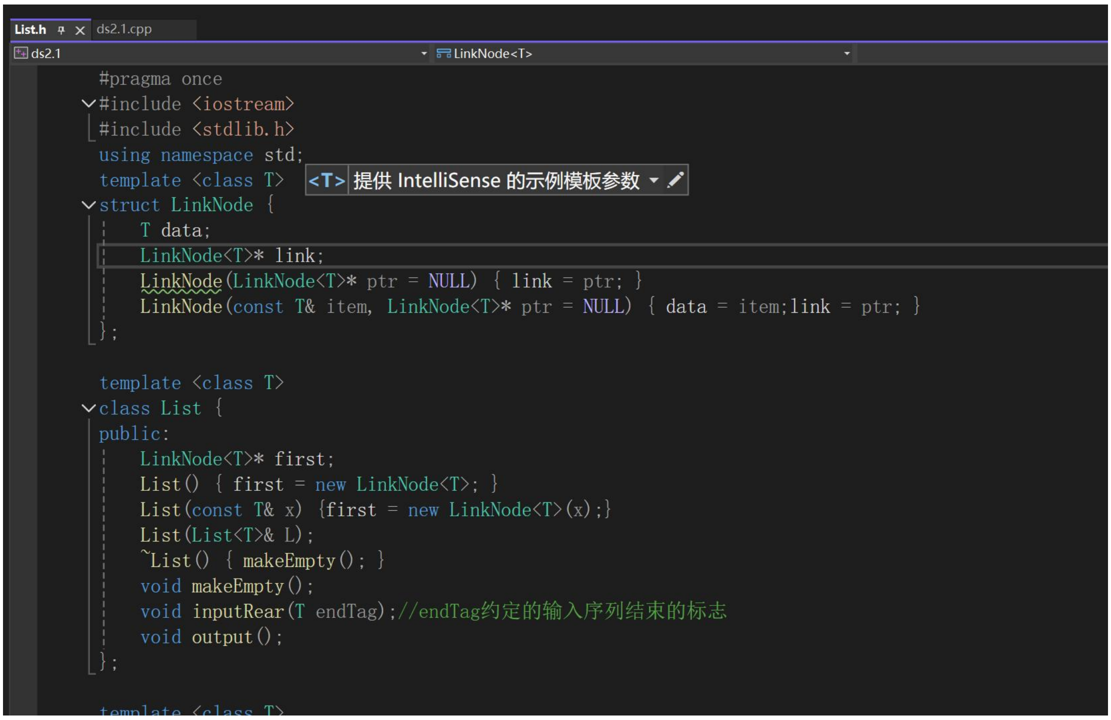
Ha=Ha→link;

```
五/3. void findmaxmin(List<int> A, int&max int&min){
    int n=A.Length(),x;
   A.getData(1,max);
   min=max;
  for(int i=2;i<=n;i++){
```

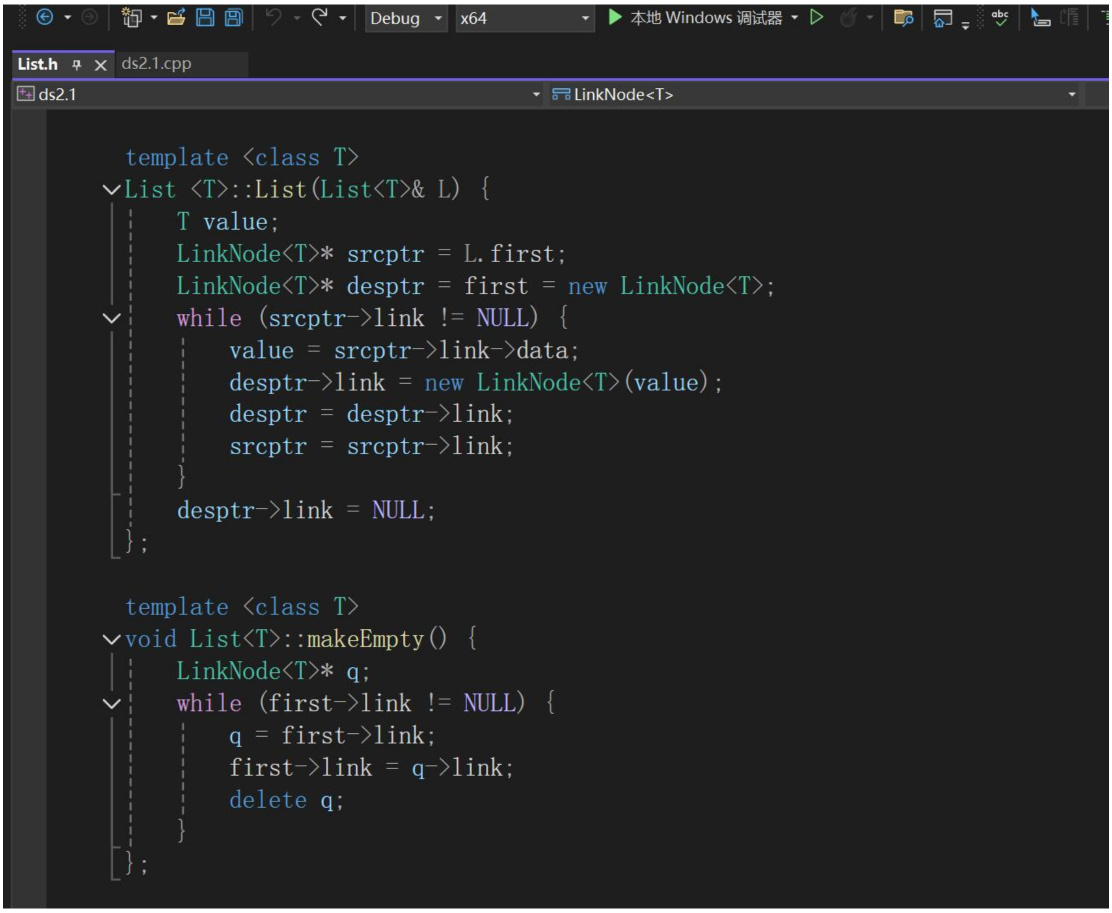


```
A.getData(i, x);
if(x>max) max=x;
else if (x< min)  min=x;}
}

五/8. void arrange (LinkNode * ha, int n){

for(int i=1;i<=n/2;i++)
{ LinkNode * p= ha;
for(int j=0; j<n; j++)  p=p->link;
LinkNode * q= ha;        {
for(int j=0;j<n-1;j++)  q=q->link;
q->link=p->link;
LinkNode * m=ha;
for(int j=0;j< 2*i-1;j++)  m=m->link;
p->link=m->link;
m->link=p;}
}
```

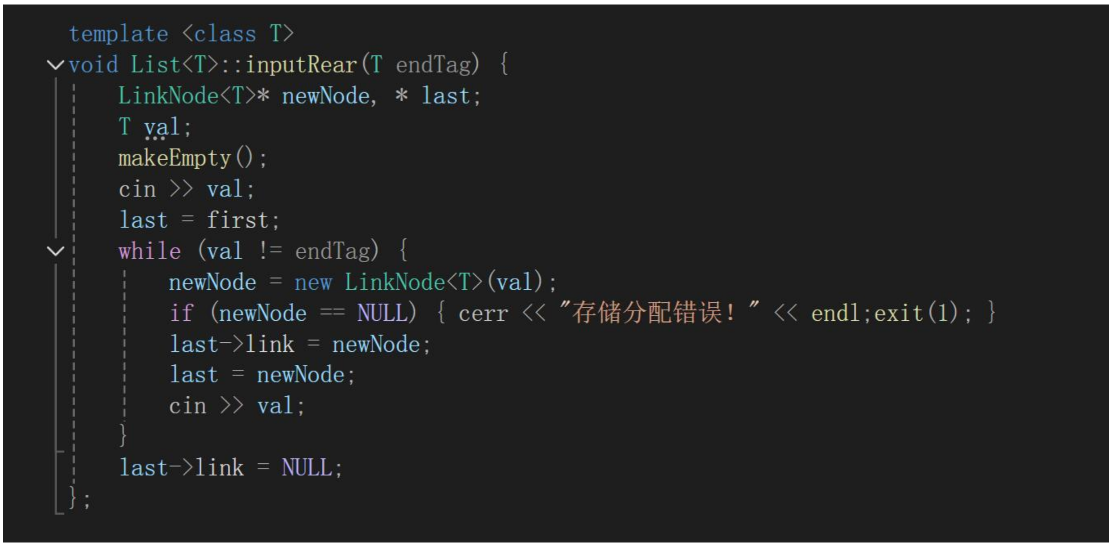
2.1



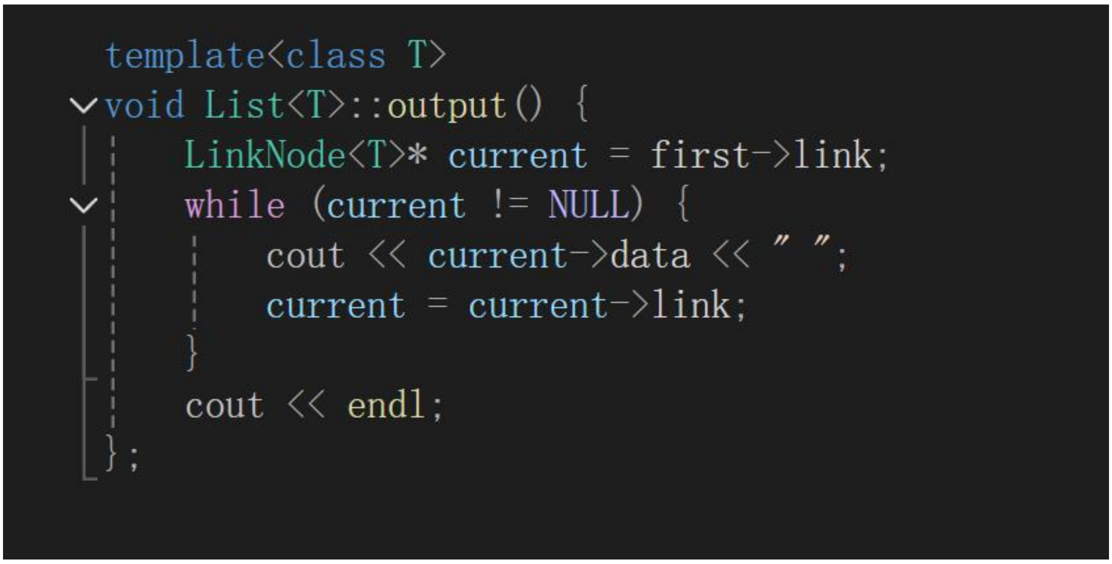
```
#pragma once
#include <iostream>
#include <stdlib.h>
using namespace std;
template <class T>
struct LinkNode {
    T data;
    LinkNode<T>* link;
    LinkNode(LinkNode<T>* ptr = NULL) { link = ptr; }
    LinkNode(const T& item, LinkNode<T>* ptr = NULL) { data = item;link = ptr; }
};

template <class T>
class List {
public:
    LinkNode<T>* first;
    List() { first = new LinkNode<T>; }
    List(const T& x) {first = new LinkNode<T>(x);}
    List(List<T>& L);
    ~List() { makeEmpty(); }
    void makeEmpty();
    void inputRear(T endTag);//endTag约定的输入序列结束的标志
    void output();
};

template <class T>
```



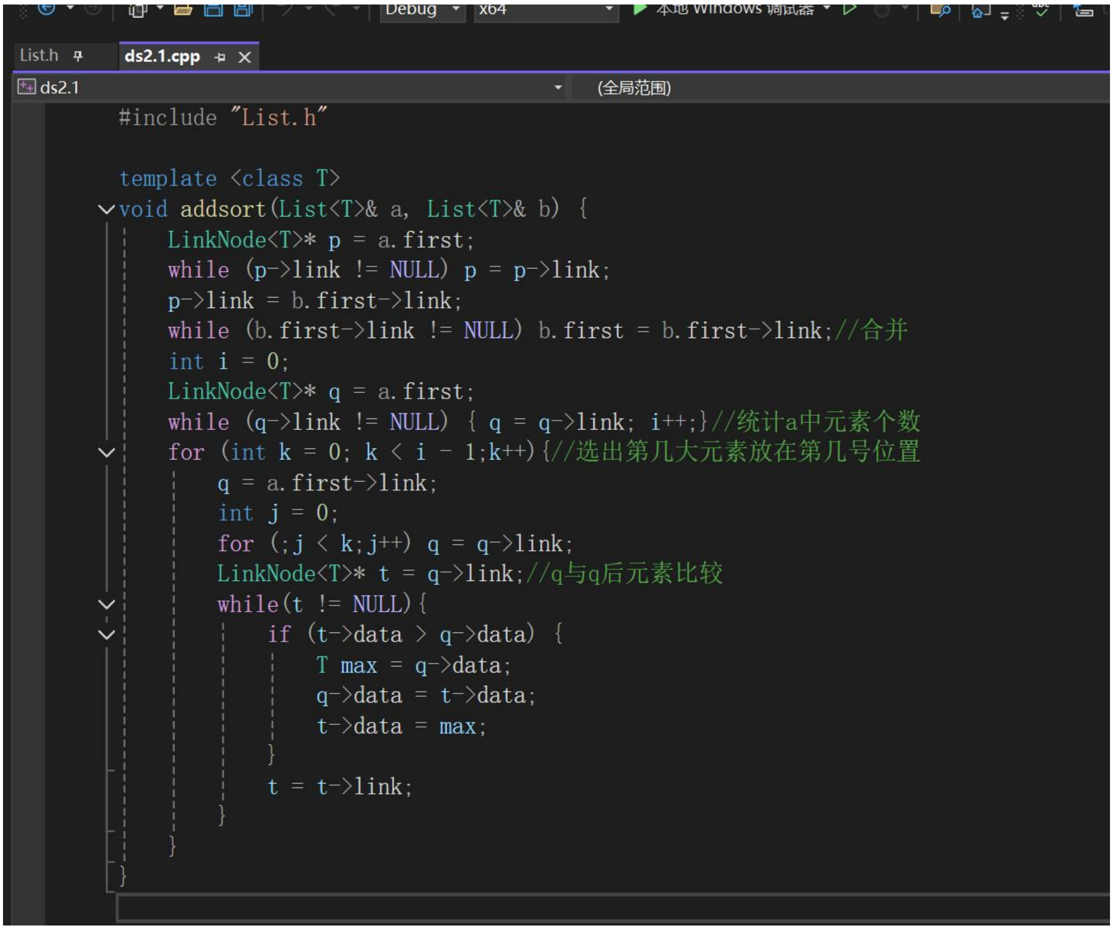
```cpp
template <class T>
List <T>::List(List<T>& L) {
    T value;
    LinkNode<T>* srcptr = L.first;
    LinkNode<T>* desptr = first = new LinkNode<T>;
    while (srcptr->link != NULL) {
        value = srcptr->link->data;
        desptr->link = new LinkNode<T>(value);
        desptr = desptr->link;
        srcptr = srcptr->link;
    }
    desptr->link = NULL;
};

template <class T>
void List<T>::makeEmpty() {
    LinkNode<T>* q;
    while (first->link != NULL) {
        q = first->link;
        first->link = q->link;
        delete q;
    }
};
```



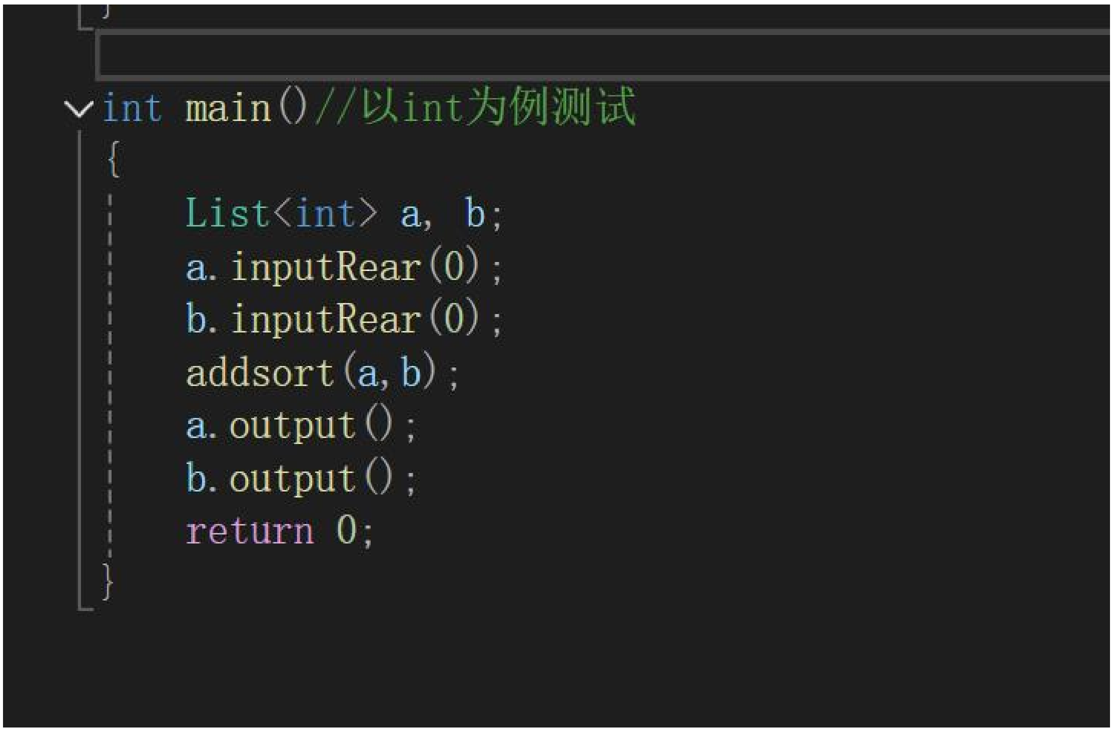
```cpp
template <class T>
void List<T>::inputRear(T endTag) {
    LinkNode<T>* newNode, * last;
    T val;
    makeEmpty();
    cin >> val;
    last = first;
    while (val != endTag) {
        newNode = new LinkNode<T>(val);
        if (newNode == NULL) { cerr << "存储分配错误！" << endl;exit(1); }
        last->link = newNode;
        last = newNode;
        cin >> val;
    }
    last->link = NULL;
};
```



```
template<class T>
void List<T>::output() {
    LinkNode<T>* current = first->link;
    while (current != NULL) {
        cout << current->data << ” ”;
        current = current->link;
    }
    cout << endl;
};
```

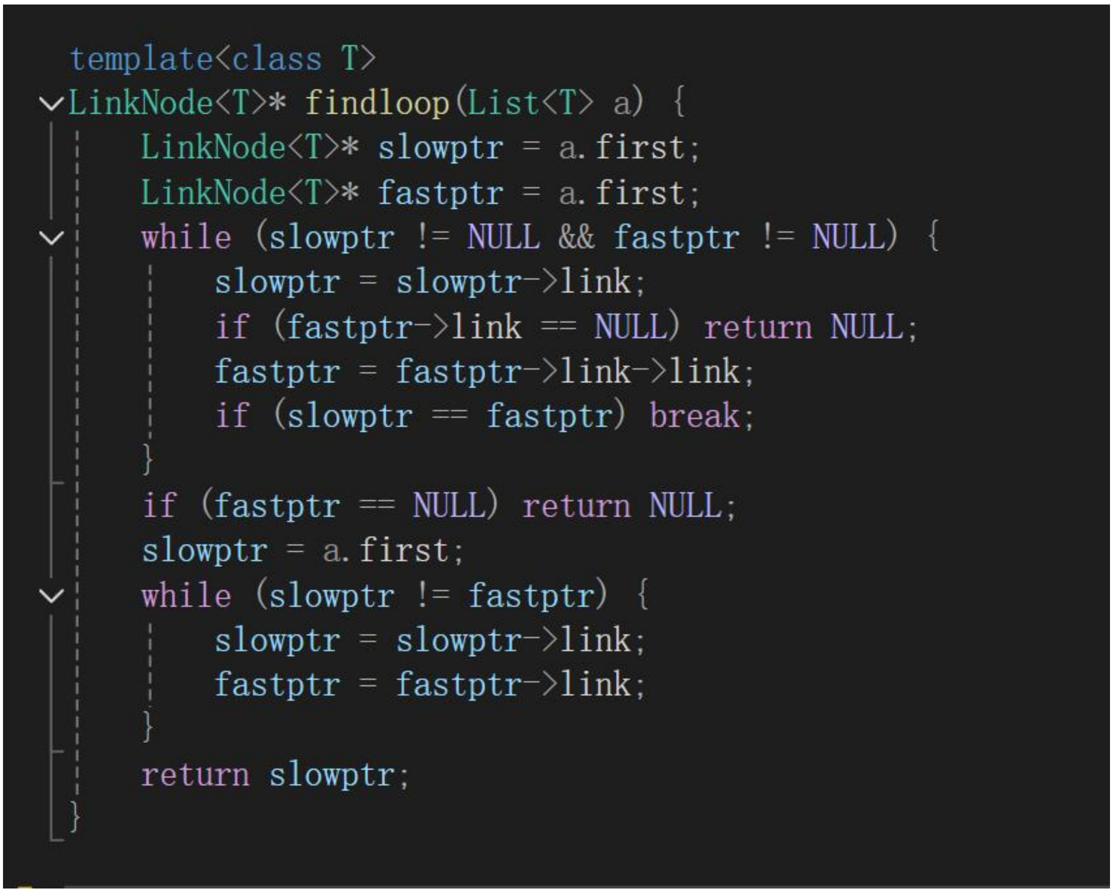


```
#include ”List.h”

template <class T>
void addsort(List<T>& a, List<T>& b) {
    LinkNode<T>* p = a.first;
    while (p->link != NULL) p = p->link;
    p->link = b.first->link;
    while (b.first->link != NULL) b.first = b.first->link;//合并
    int i = 0;
    LinkNode<T>* q = a.first;
    while (q->link != NULL) { q = q->link; i++;}//统计a中元素个数
    for (int k = 0; k < i - 1;k++){//选出第几大元素放在第几号位置
        q = a.first->link;
        int j = 0;
        for (;j < k;j++) q = q->link;
        LinkNode<T>* t = q->link;//q与q后元素比较
        while(t != NULL){
            if (t->data > q->data) {
                T max = q->data;
                q->data = t->data;
                t->data = max;
            }
            t = t->link;
        }
    }
}
```



```
}

int main()//以int为例测试
{
    List<int> a,  b;
    a.inputRear(0);
    b.inputRear(0);
    addsort(a,b);
    a.output();
    b.output();
    return 0;
}
```

2.2



```
 template<class T>
LinkNode<T>* findloop(List<T> a) {
    LinkNode<T>* slowptr = a.first;
    LinkNode<T>* fastptr = a.first;
    while (slowptr != NULL && fastptr != NULL) {
        slowptr = slowptr->link;
        if (fastptr->link == NULL) return NULL;
        fastptr = fastptr->link->link;
        if (slowptr == fastptr) break;
    }
    if (fastptr == NULL) return NULL;
    slowptr = a.first;
    while (slowptr != fastptr) {
        slowptr = slowptr->link;
        fastptr = fastptr->link;
    }
    return slowptr;
}
```