**源代码：**

第四题：

//miniNode project\_Task4.

//Created by Yule, Oct\_13\_2015

//=============================================================================

#include<iostream>

using namespace std;

template <typename T>

class node

{

public:

T nodeValue; // data held by the node

node<T> \*next; // next node in the list

node() : next(NULL) {}

node(const T& item, node<T> \*nextNode = NULL) : nodeValue(item), next(nextNode) {}

};

int main()

{

// data input section.

int num, temp, max;

node<int> \*first = NULL, \*p[2], \*ser;

cout<<"Please enter the sum number of the data: ";

cin>>num;

cout<<"Please enter the data: ";

// using the subscript complementation way to avoid

// transmitting points addresses and accelerate the

// speed.

cin>>temp;

first = new node<int>(temp, NULL);

p[0] = first;

for (int i = 1; i<num; i++){

cin>>temp;

p[i%2] = new node<int>(temp, NULL);

p[(i-1)%2]->next = p[i%2];

}

// output the original inputted data.

cout<<"The data inputted: ";

ser = first;

while(ser!=NULL){

cout<<ser->nodeValue<<' ';

ser=ser->next;

}

cout<<endl;

// judge section

// fit the variable 'max' with the value got, and compare with the next value.

// by setting the boolean variable to determine whether to meet the requirements.

bool ju = 1;

ser = first;

max = first->nodeValue;

while(ser!=NULL){

if(ser->nodeValue < max){

ju = 0;

}

else

max = ser->nodeValue;

ser=ser->next;

}

cout<<"This is"<<(ju? "" : " not")<<" sorted by ascending order."<<endl;

return 0;

}

第七题：

//miniNode project\_Task7.

//Created by Yule, Oct\_14\_2015

//=============================================================================

#include<iostream>

using namespace std;

template <typename T>

class node

{

public:

T nodeValue; // data held by the node

node<T> \*next; // next node in the list

node() : next(NULL) {}

node(const T& item, node<T> \*nextNode = NULL) : nodeValue(item), next(nextNode) {}

};

int main()

{

// data input section

int num, temp, max, place;

node<int> \*first = NULL, \*p[2], \*ser;

cout<<"Please enter the sum number of the data: ";

cin>>num;

cout<<"Please enter the data: ";

// using the subscript complementation way to avoid

// transmitting points addresses and accelerate the

// speed.

cin>>temp;

first = new node<int>(temp, NULL);

p[0] = first;

for (int i = 1; i<num; i++){

cin>>temp;

p[i%2] = new node<int>(temp, NULL);

p[(i-1)%2]->next = p[i%2];

}

// output the original inputted data.

cout<<"The data inputted: ";

ser = first;

while(ser!=NULL){

cout<<ser->nodeValue<<' ';

ser=ser->next;

}

cout<<endl;

// insert section

// input the number of requests after the node N.

cout<<"Please enter the node to insert: ";

cin>>place;

// judging whether the input node exists.

if (place > num)

cout<<"The location of the input is incorrect!"<<endl;

// ensure the presence of the input node, enter the data .

else{

node<int> \*insert;

cout<<"Please enter the data: ";

cin>>temp;

// if want to enter the node in the first or the last place,

// take out the analysis alone.

// Node in the first place. It means that insert the data in a

// new created space, which links to the original first node's next space

// and let the first point points to this new space.

if (place == 1){

insert = new node<int>(temp, first->next);

first->next = insert;

}

// Node in the final place. It means that insert the data in a

// new created space, which links to NULL space

// and let the last NULL point points to this new space.

else if (place == num){

ser = first;

while(ser->next!=NULL)

ser=ser->next;

insert = new node<int>(temp, NULL);

ser->next = insert;

}

// when the node is among the list, it's needed to create two points:

// Precursor pointer 'pre' and Subsequent pointer 'sub'.

// Use pre to find the node, and let the next node become sub.

// Create a new space to insert the data. Let pre points to this space and

// this space points to sub.

else{

node<int> \*pre, \*sub;

pre = first;

for (int i=0; i<place-1; i++){

pre = pre->next;

}

sub = pre->next;

insert = new node<int>(temp, sub);

pre->next = insert;

}

//Output modified data

cout<<"Output modified data: ";

ser = first;

while(ser!=NULL){

cout<<ser->nodeValue<<' ';

ser=ser->next;

}

cout<<endl;

}

return 0;

}

第十二题：

//miniNode project\_Task12.

//Created by Yule, Oct\_14\_2015

//==============================================================================

#include<iostream>

using namespace std;

template <typename T>

class node

{

public:

T nodeValue; // data held by the node

node<T> \*next; // next node in the list

node() : next(NULL) {}

node(const T& item, node<T> \*nextNode = NULL) : nodeValue(item), next(nextNode) {}

};

int main()

{

// data input section.

int num, temp, place;

node<int> \*first = NULL, \*newNode, \*p[2];

cout<<"Please enter the sum number of the data: ";

cin>>num;

cout<<"Please enter the data: ";

// using the subscript complementation way to avoid

// transmitting points addresses and accelerate the

// speed.

cin>>temp;

first = new node<int>(temp, NULL);

p[0] = first;

for (int i = 1; i<num; i++){

cin>>temp;

p[i%2] = new node<int>(temp, NULL);

p[(i-1)%2]->next = p[i%2];

}

// output the original inputted data.

cout<<"The data inputted: ";

node<int> \*ser;

ser = first;

while(ser != NULL){

cout<<ser->nodeValue<<' ';

ser=ser->next;

}

cout<<endl;

// Reverse the original nodes,

// here only change of the pointers' directions to achieve the goal.

node<int> \*pre, \*sub;

// let pre equals to first

pre = first;

// if there doesn't exist any node, pick out to discuss.

if (pre == NULL)

cout<<"Output modified data after backward: "<<endl;

else{

// save the second node to ser.

ser = pre->next;

// The original first become the last after reverse,

// so let it's next point to NULL.

pre->next = NULL;

// now pre is the new list's first. And in this new list,

// there is only one node, pre.

// go over the original list, start from the second one.

while(ser){

// save the current node's next to sub.

sub = ser->next;

// change the pre to current node ser's subsequent.

ser->next = pre;

// now pre is the new list's first.

pre = ser;

// assign the original list to ser.

ser = sub;

}

// after circulation, pre is the new first.

// assign to first.

first = pre;

}

//Output modified data

cout<<"Output modified data after backward: " ;

ser = first;

while(ser != NULL){

cout<<ser->nodeValue<<' ';

ser=ser->next;

}

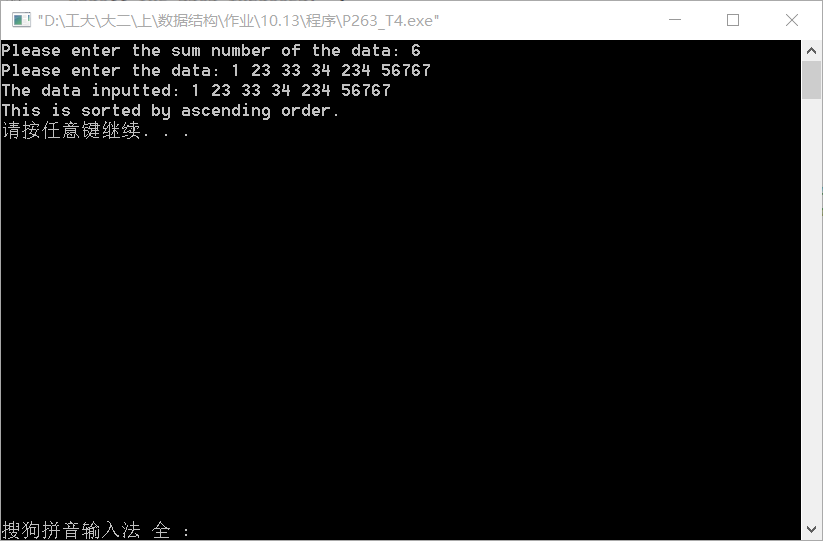
cout<<endl;

return 0;

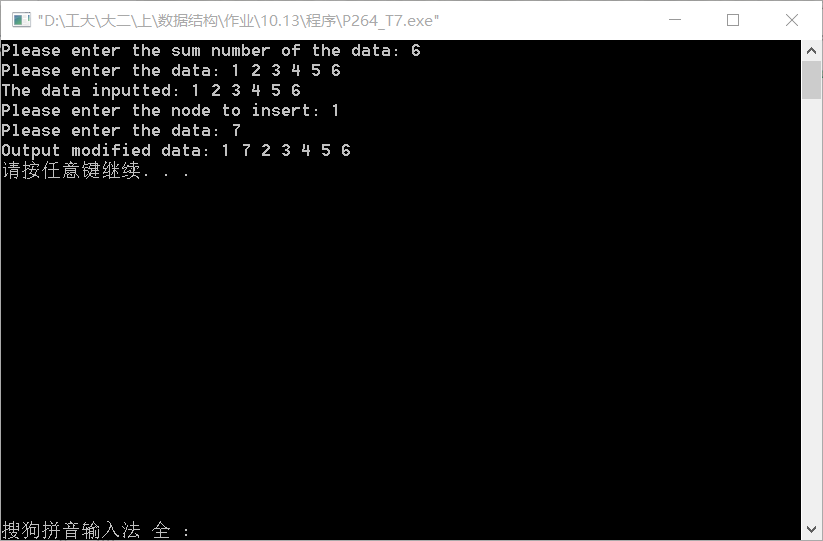
}

**运行截图：**

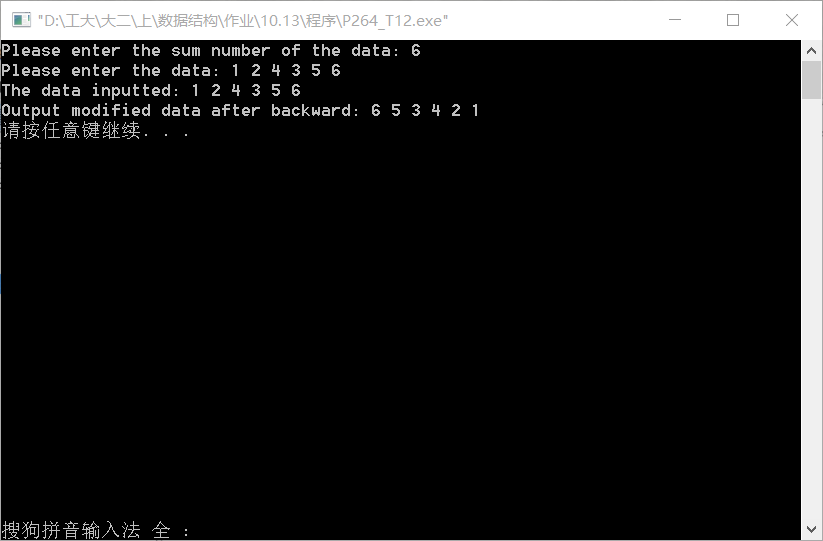
第四题：



第七题：



第十二题：



**分析：**

第四题：输入后只需要判断后面的数据是否比前一个大就可以了。

第七题：插入要分三种情况来看，如果是在第一位节点后插入，只需要将插入节点的指针指向第二位，开始节点的指针指向新建节点；如果是在最后一位节点后插入，只需要将最后的指针指向新建的节点，让新建的节点指向NULL；如果是在其中的某一位插入，就需要用到前驱指针和后继指针。新建节点指向后继指针的位置，前驱指针指向新建节点即可。

第十二题：输入原始数据后，先判断节点个数，然后用类似反向建立链表的方式，借用辅助的指针，将原来链表指针方向改为指向前面，最后将原来尾指针赋值给原来首指针即可。