**1001: 顺序表（1）**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 165  Solved: 53  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1001)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1001)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1001)]

**Description**

**已知顺序表类的定义如下，实现各个成员函数。主函数中输入数据（以0结束）利用Insert函数依次将数据插入到表的1号位置，利用DispList按照逻辑次序输出表中元素，再输入一个要查找的元素，利用查找函数Locate查找其在表中的位置，最后利用Reverse函数将数据逆序，再利用DispList输出。**

**template <class T>**

**class SeqList{**

**public:**

**SeqList(); //构造函数，将表置空**

**~SeqList(){} //析构**

**int Locate(T x); //查找元素x在表中的位置，成功返回x在表中的位置，否则返回0**

**void Insert(int i, T x); //在表的i位置插入元素x**

**void Reverse(); //逆序表中数据**

**void DispList();**

**private:**

**T data[MaxSize];    //存储元素**

**int length;              //顺序表实际长度**

**};**

**Input**

**输入样例说明：例如输入数据为：1 2 3 4 5 6 0 3，即将1,2,3,4,5,6插入表中的1号位置，得到逻辑次序为6,5,4,3,2,1的顺序表，3为在表中待查找的数据，3的位置为4。**

**输入：1 2 3 4 5 6 0 3**

**输出：**

**The length:6  
The elements:  
6 5 4 3 2 1   
Found position:4  
The length:6  
The elements:  
1 2 3 4 5 6**

**若查找的数据不存在，则输出：No found，例如下例**

**输入：1 2 3 4 5 6 0 9**

**输出：**

**The length:6  
The elements:  
6 5 4 3 2 1   
No found  
The length:6  
The elements:  
1 2 3 4 5 6**

**Output**

#include<iostream>

using namespace std;

const int MaxSize=100;

template<class T>

class SeqList{

public:

SeqList();

~SeqList()

{

}

int Locate(T x);

void Insert(int i,T x);

void Reverse();

void DispList();

private:

T data[MaxSize];

int length;

};

template<class T>

SeqList<T>::SeqList()

{

length=0;

}

template<class T>

int SeqList<T>::Locate(T x)

{

int i;

for(i=0;i<length;i++)

if(data[i]==x)

{cout<<"Found position:"<<i+1<<endl;break;}

else if(i==length-1) cout<<"No found"<<endl;

return 0;

}

template<class T>

void SeqList<T>::Insert(int i,T x)

{

int z;

for(z=length;z>0;z--)

data[z]=data[z-1];

data[i]=x;

length++;

}

template<class T>

void SeqList<T>::Reverse()

{

int i;

T x;

for(i=0;i<length/2.0;i++)

{x=data[i];

data[i]=data[length-i-1];

data[length-i-1]=x;}

}

template<class T>

void SeqList<T>::DispList()

{ int i;

cout<<"The length:"<<length<<endl;

cout<<"The elements:"<<endl;

for(i=0;i<length;i++)

cout<<data[i]<<" ";

cout<<endl;

}

int main()

{

int x;

SeqList<int>R;

while(cin>>x)

{

if(x==0) break;

R.Insert(0,x);

}

R.DispList();

cin>>x;

R.Locate(x);

R.Reverse();

R.DispList();

return 0;

}

**1002: 顺序表（2）**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 68  Solved: 45  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1002)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1002)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1002)]

**Description**

**已知一个有序顺序表类SortList及main函数的部分代码如下，请完成SortList类的成员函数Insert和DispList，得到对应的运行结果，勿改动main函数。**

**//有序表类**

**template <class T>**

**class SortList{**

**public:**

**SortList(){length=0;}**

**~SortList(){}**

**void Insert(T x);    //有序表的插入，使序列仍有序**

**void DispList();      //输出表**

**private:**

**T data[MaxSize]; //存储元素**

**int length;              //顺序表实际长度**

**};**

**//构造有序表A：函数声明**

**template <class T>**

**void CreateSort(SortList <T> &A);**

**int main(){**

**SortList <int> A;**

**//生成一个有序表A**

**CreateSort(A);**

**SortList <char> B;**

**CreateSort(B);**

**A.DispList();**

**B.DispList();**

**return 0;**

**}**

**//构造有序表A：函数定义**

**template <class T>**

**void CreateSort(SortList <T> &A){**

**int i,n;**

**T x;**

**cin>>n;**

**for (i=1;i<=n;i++){**

**cin>>x;**

**try{**

**A.Insert(x);**

**}**

**catch(const char \*wrong){**

**cout<<wrong<<endl;**

**}**

**}**

**}**

**Input**

**数据输入说明：下列两行输入分别代表创建A表和B表，元素分别为整型和字符型，每行输入中第一个值为元素个数，之和为待插入的数据。**

**5 3 24 23 2 6  
4 abnf**

**Output**

**输出数据用空格隔开**

**Sample Input**

**5 3 24 23 2 6**

**4 abnf**

**Sample Output**

**The length:5**

**The elements:**

**2 3 6 23 24**

**The length:4**

**The elements:**

**a b f n**

#include<iostream>

using namespace std;

const int MaxSize=100;

template<class T>

class SortList{

public:

SortList();

~SortList()

{}

void Insert(T x);

void DispList();

private:

T data[MaxSize];

int length;

};

template<class T>

SortList<T>::SortList()

{

length=0;

}

template<class T>

void SortList<T>::Insert(T x)

{

int i,z;

for(i=0;i<length&&data[i]<x;i++);

for(z=length;z>i;z--)

data[z]=data[z-1];

data[i]=x;

length++;

}

template<class T>

void SortList<T>::DispList()

{ int i;

cout<<"The length:"<<length<<endl;

cout<<"The elements:"<<endl;

for(i=0;i<length;i++)

cout<<data[i]<<" ";

cout<<endl;

}

template <class T>

void CreateSort(SortList <T> &A);

int main(){

SortList <int> A;

//生成一个有序表A

CreateSort(A);

SortList <char> B;

CreateSort(B);

A.DispList();

B.DispList();

return 0;

}

template <class T>

void CreateSort(SortList <T> &A)

{

int i,n;

T x;

cin>>n;

for (i=1;i<=n;i++){

cin>>x;

try{

A.Insert(x);

}

catch(const char \*wrong){

cout<<wrong<<endl;

}

}

}

**1003: 顺序表（3）**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 55  Solved: 44  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1003)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1003)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1003)]

**Description**

**请在上题SortList类的基础上添加成员函数Merge，实现两个有序表的合并。main代码已知，请勿改动。**

//有序表类

template <class T>

class SortList{

       public:

              SortList(){length=0;}

              ~SortList(){}

              void Insert(T x);    //有序表的插入，使序列仍有序

            void DispList();      //输出表

           /\*成员函数Merge实现两个有序表的合并，使序列仍有序，

              将A表和B表合并到当前类中，要求：A表，B表的元素保持不变\*/

              void Merge(SortList <T> &A,SortList <T> &B);

       private:

              T data[MaxSize]; //存储元素

          int length;              //顺序表实际长度

};

//构造有序表A：函数声明

template <class T>

void CreateSort(SortList <T> &A);

int main(){

       SortList <int> A,B,C;

       //生成一个有序表A

       CreateSort(A);

       //生成一个有序表B

       CreateSort(B);

       try{

              C.Merge(A,B);//合并A,B表为C表

       }

    catch(const char\* wrong){

              cout << wrong;    //如失败提示失败信息

       }

      A.DispList();

       B.DispList();

       C.DispList(); //显示合并后的结果

       return 0;

}

//构造有序表A：函数定义

template <class T>

void CreateSort(SortList <T> &A){

       int i,n;

       T x;

       cin>>n;

       for (i=1;i<=n;i++){

              cin>>x;

              try{

                     A.Insert(x);

              }

              catch(char \*wrong){

                     cout<<wrong<<endl;

              }

       }

}

**Input**

**Output**

**Sample Input**

5 4 24 2 42 3

6 78 36 34 24 64 43

**Sample Output**

The length:5

The elements:

2 3 4 24 42

The length:6

The elements:

24 34 36 43 64 78

The length:11

The elements:

2 3 4 24 24 34 36 42 43 64 78

#include<iostream>

using namespace std;

const int MaxSize=100;

template<class T>

class SortList{

public:

SortList();

~SortList()

{}

void Insert(T x);

void DispList();

void Merge(SortList<T>&A,SortList<T>&B);

private:

T data[MaxSize];

int length;

};

template<class T>

void SortList<T>::Merge(SortList<T>&A,SortList<T>&B)

{

int i;

for(i=0;i<A.length;i++)

Insert(A.data[i]);

for(i=0;i<B.length;i++)

Insert(B.data[i]);

}

template<class T>

SortList<T>::SortList()

{

data[0]=10000;

length=0;

}

template<class T>

void SortList<T>::Insert(T x)

{

int i,z;

if(length==MaxSize) throw"wrong";

else

{for(i=0;i<length&&data[i]<x;i++);

for(z=length;z>i;z--)

data[z]=data[z-1];

data[i]=x;

length++;}

}

template<class T>

void SortList<T>::DispList()

{ int i;

cout<<"The length:"<<length<<endl;

cout<<"The elements:"<<endl;

for(i=0;i<length;i++)

cout<<data[i]<<" ";

cout<<endl;

}

template <class T>

void CreateSort(SortList <T> &A);

int main(){

SortList <int> A,B,C;

//生成一个有序表A

CreateSort(A);

//生成一个有序表B

CreateSort(B);

try{

C.Merge(A,B);//合并A,B表为C表

}

catch(const char\* wrong){

cout << wrong; //如失败提示失败信息

}

A.DispList();

B.DispList();

C.DispList(); //显示合并后的结果

return 0;

}

template <class T>

void CreateSort(SortList <T> &A)

{

int i,n;

T x;

cin>>n;

for (i=1;i<=n;i++){

cin>>x;

try{

A.Insert(x);

}

catch(const char \*wrong){

cout<<wrong<<endl;

}

}

}

## 1004: 单向链表（1）

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 397  Solved: 222  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1004)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1004)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1004)]

**Description**

**已知链表类的定义如下，实现各个成员函数。主函数中输入数据（以0结束）利用Insert函数依次将数据插入到表的1号位置，利用DispList按照逻辑次序输出表中元素，再输入一个要查找的元素，利用查找函数Locate查找其在表中的位置，最后利用Reverse函数将数据逆序，再利用DispList输出。**

template <class T>

class LinkList

{

public:

LinkList( ); //建立只有头结点的空链表

~LinkList();             //析构函数

int Length();          //求单链表的长度

int Locate(T x);       //求单链表中值为x的元素序号

void Insert(int i, T x);   //在单链表中第i个位置插入元素值为x的结点

void Reverse( );        //reverse list

void DispList( );           //遍历单链表，按序号依次输出各元素

private:

Node<T> \*first; //单链表的头指针

};

输入样例说明：

例如输入数据为：1 2 3 4 5 6 0 3，即将1,2,3,4,5,6插入表中的1号位置，得到逻辑次序为6,5,4,3,2,1的顺序表，3为在表中待查找的数据，3的位置为4。

若输入：1 2 3 4 5 6 0 13

13在表中无，则输出：No found

即输出结果：

The length:6

The elements:  
6 5 4 3 2 1   
No found

The length:6  
The elements:  
1 2 3 4 5 6

## Input

**Output**

**Sample Input**

1 2 3 4 5 6 0 3

**Sample Output**

The length:6

The elements:

6 5 4 3 2 1

Found position:4

The length:6

The elements:

1 2 3 4 5 6

#include <iostream>

using namespace std;

template <class T>

struct Node

{

    T data;

    Node<T> \*next;

};

template <class T>

class LinkSortList

{

 public:

    LinkSortList( ); //建立只有头结点的空链表

    ~LinkSortList();             //析构函数

    void Insert(T x);   //在有序单链表中插入元素x使序列仍有序

    int Length();      //求表长

    void DispList( );     //遍历有序单链表，按序号依次输出各元素

    void Merge(LinkSortList<T> &A,LinkSortList<T> &B);

 private:

   Node<T> \*first; //单链表的头指针

};

template <class T>

LinkSortList<T>::LinkSortList()

{

    first=new Node<T>;

    first->next=NULL;

}

template <class T>

LinkSortList<T>::~LinkSortList()

{

    Node <T> \*p;

    for(  ; first!=NULL; )

    {   p=first;

        first=first->next;

        delete p;

    }

}

template <class T>

void LinkSortList<T>::Insert(T x)

{

    Node <T> \*p,\*s;

    s=new Node<T>;

    s->data=x;

    for( p=first ; p->next!=NULL&& s->data > p->next->data ; p=p->next );

    s->next=p->next;

    p->next=s;

}

template <class T>

int LinkSortList<T>::Length()

{

    Node<T> \*p;

    int i;

    for(p=first->next ,i=0; p!=NULL ; p=p->next ,i++);

    return i;

}

template <class T>

void LinkSortList<T>::Merge(LinkSortList<T> &A,LinkSortList<T> &B)

{

    Node<T> \*p,\*q,\*s;

    p=A.first->next;

    A.first->next=NULL;

    q=B.first->next;

    B.first->next=NULL;

    for(s=first ; p&&q ; s->next=NULL )

        if(p->data<=q->data)

        {

            s->next=p;

            p=p->next;

            s=s->next;

        }

        else

        {

            s->next=q;

            q=q->next;

            s=s->next;

        }

    if(q)

    {

        s->next=q;

    }

    if(p)

    {

        s->next=p;

    }

}

template <class T>

void LinkSortList<T>::DispList()

{

    cout<<"The length:"<<Length()<<endl;

    cout<<"The elements:";

    Node<T> \*p;

    for(p=first->next; p!=NULL ; p=p->next)

        cout<<p->data<<" ";

    cout<<endl;

}

int main( )

{

 LinkSortList<int> A,B,C;

 int x;

     while(1)

     {

            cin>>x;

            if(!x)break;

            A.Insert(x);

     }

    while(1)

    {

             cin>>x;

             if(!x) break;

             B.Insert(x);

       }

       A.DispList();

       B.DispList();

       C.Merge(A,B);

       C.DispList();

        A.DispList();

       B.DispList();

       return 0;

}

## 1005: 单向链表（2）

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 238  Solved: 183  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1005)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1005)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1005)]

## Description

**已知一个有序链表类LinkSortList及main函数的部分代码如下，请完成LinkSortList类的成员函数，得到对应的运行结果，勿改动main函数。**

**//有序表类**

template <class T>

class LinkSortList

{

 public:

    LinkSortList( ); //建立只有头结点的空链表

    ~LinkSortList();             //析构函数

    void Insert(T x);   //在有序单链表中插入元素x使序列仍有序

    int Length();      //求表长

    int DeleteOdd();     //在单链表中删除所有奇数，返回值为奇数个数

    void DispList( );     //遍历有序单链表，按序号依次输出各元素

 private:

   Node<T> \*first; //单链表的头指针

};

int main( ){  
 LinkSortList<int> sa;  
 int x;  
 while(1)  
 {  
  cin>>x;  
        if(!x)break;  
  sa.Insert(x);    
 }  
    sa.DispList();  
 int count=sa.DeleteOdd();  
 cout<<"Count of deleted odds:"<<count<<endl;  
 sa.DispList();  
 return 0;  
}

**Input**

**Output**

**Sample Input**

**42 5 32 56 34 7 233 1 0**

**Sample Output**

**The length:8**

**The elements:1 5 7 32 34 42 56 233**

**Count of deleted odds:4**

**The length:4**

**The elements:32 34 42 56**

#include <iostream>

using namespace std;

template <class T>

struct Node

{

    T data;

    Node<T> \*next;

};

template <class T>

class LinkSortList

{

 public:

    LinkSortList( ); //建立只有头结点的空链表

    ~LinkSortList();             //析构函数

    void Insert(T x);   //在有序单链表中插入元素x使序列仍有序

    int Length();      //求表长

    int DeleteOdd();     //在单链表中删除所有奇数，返回值为奇数个数

    void DispList( );     //遍历有序单链表，按序号依次输出各元素

 private:

   Node<T> \*first; //单链表的头指针

};

template <class T>

LinkSortList<T>::LinkSortList()

{

    first=new Node<T>;

    first->next=NULL;

}

template <class T>

LinkSortList<T>::~LinkSortList()

{

    Node <T> \*p;

    for(  ; first!=NULL; )

    {   p=first;

        first=first->next;

        delete p;

    }

}

template <class T>

void LinkSortList<T>::Insert(T x)

{

    Node <T> \*p,\*s;

    s=new Node<T>;

    s->data=x;

    for( p=first ; p->next!=NULL&& s->data > p->next->data ; p=p->next );

    s->next=p->next;

    p->next=s;

}

template <class T>

int LinkSortList<T>::Length()

{

    Node<T> \*p;

    int i;

    for(p=first->next ,i=0; p!=NULL ; p=p->next ,i++);

    return i;

}

template <class T>

int LinkSortList<T>::DeleteOdd()

{

    int i=0;

    Node <T> \*p,\*s;

    for(p=first ; p->next!=NULL ; )

        if(p->next->data%2==1) {

            s=p->next;

            p->next=p->next->next;

            i++;

            delete s;

        }

        else p=p->next;

        return i;

}

template <class T>

void LinkSortList<T>::DispList()

{

    cout<<"The length:"<<Length()<<endl;

    cout<<"The elements:";

    Node<T> \*p;

    for(p=first->next; p!=NULL ; p=p->next)

        cout<<p->data<<" ";

    cout<<endl;

}

int main( ){

 LinkSortList<int> sa;

 int x;

 while(1)

 {

  cin>>x;

        if(!x)break;

  sa.Insert(x);

 }

    sa.DispList();//while(1);

    //cout<<"A"<<endl;

 int count=sa.DeleteOdd();

 cout<<"Count of deleted odds:"<<count<<endl;

 sa.DispList();

 return 0;

}

**1006: 单向链表（3）**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 174  Solved: 150  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1006)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1006)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1006)]

**Description**

**1、 请在LinkSortList类的基础上添加成员函数Merge，实现将两个有序表A，B的合并为表C，C中序列仍有序。要求利用A，B中的结点进行合并，合并后A，B置为空表。main代码已知，请勿改动。**

//有序表类

template <class T>

class LinkSortList

{

 public:

    LinkSortList( ); //建立只有头结点的空链表

    ~LinkSortList();             //析构函数

      void Insert(T x);        //sort insert

       int Length();

    void DispList( );           //遍历单链表，按序号依次输出各元素

    void Merge(LinkSortList<T> &A, LinkSortList<T> &B); //合并A,B到当前类

 private:

   Node<T> \*first; //单链表的头指针

};

int main( ){

       LinkSortList <int> A,B,C;

       int x;

       while(1){

              cin>>x;

              if(!x) break;

              A.Insert(x);

       }

       while(1){

              cin>>x;

              if(!x) break;

              B.Insert(x);

       }

       A.DispList();

       B.DispList();

       C.Merge(A,B);

       C.DispList();

    A.DispList();

       B.DispList();

       return 0;

}

**Input**

**Output**

**Sample Input**

342 4521 521 4 21 3 4 21 4 0

234 5 3 25 5236 34 65 23 34 0

**Sample Output**

The length:9

The elements:3 4 4 4 21 21 342 521 4521

The length:9

The elements:3 5 23 25 34 34 65 234 5236

The length:18

The elements:3 3 4 4 4 5 21 21 23 25 34 34 65 234 342 521 4521 5236

The length:0

The elements:

The length:0

The elements:

#include <iostream>

using namespace std;

template <class T>

struct Node

{

T data;

Node<T> \*next;

};

template <class T>

class LinkSortList

{

public:

LinkSortList( ); //建立只有头结点的空链表

~LinkSortList(); //析构函数

void Insert(T x); //在有序单链表中插入元素x使序列仍有序

int Length(); //求表长

void DispList( ); //遍历有序单链表，按序号依次输出各元素

void Merge(LinkSortList<T> &A,LinkSortList<T> &B);

private:

Node<T> \*first; //单链表的头指针

};

template <class T>

LinkSortList<T>::LinkSortList()

{

first=new Node<T>;

first->next=NULL;

}

template <class T>

LinkSortList<T>::~LinkSortList()

{

Node <T> \*p;

for( ; first!=NULL; )

{ p=first;

first=first->next;

delete p;

}

}

template <class T>

void LinkSortList<T>::Insert(T x)

{

Node <T> \*p,\*s;

s=new Node<T>;

s->data=x;

for( p=first ; p->next!=NULL&& s->data > p->next->data ; p=p->next );

s->next=p->next;

p->next=s;

}

template <class T>

int LinkSortList<T>::Length()

{

Node<T> \*p;

int i;

for(p=first->next ,i=0; p!=NULL ; p=p->next ,i++);

return i;

}

template <class T>

void LinkSortList<T>::Merge(LinkSortList<T> &A,LinkSortList<T> &B)

{

Node<T> \*p,\*q,\*s;

p=A.first->next;

A.first->next=NULL;

q=B.first->next;

B.first->next=NULL;

for(s=first ; p&&q ; s->next=NULL )

if(p->data<=q->data)

{

s->next=p;

p=p->next;

s=s->next;

}

else

{

s->next=q;

q=q->next;

s=s->next;

}

if(q)

{

s->next=q;

}

if(p)

{

s->next=p;

}

}

template <class T>

void LinkSortList<T>::DispList()

{

cout<<"The length:"<<Length()<<endl;

cout<<"The elements:";

Node<T> \*p;

for(p=first->next; p!=NULL ; p=p->next)

cout<<p->data<<" ";

cout<<endl;

}

int main( )

{

LinkSortList<int> A,B,C;

int x;

while(1)

{

cin>>x;

if(!x)break;

A.Insert(x);

}

while(1)

{

cin>>x;

if(!x) break;

B.Insert(x);

}

A.DispList();

B.DispList();

C.Merge(A,B);

C.DispList();

A.DispList();

B.DispList();

return 0;

}

**1007: 单向链表（4）**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 167  Solved: 133  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1007)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1007)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1007)]

**Description**

**已知单向链表类，添加Divide函数，实现将链表分解为奇数表A和偶数表B（要求分解出的结点按照头插法的规则插入到链表的头结点之后，即元素序列与原表的序列反向）。主函数及类的定义如下（不允许改动）**

template <class T>

class LinkList

{

 public:

    LinkList( ); //建立只有头结点的空链表

   ~LinkList();             //析构函数

    int Length();          //求单链表的长度

    void Insert(int i, T x);   //在单链表中第i个位置插入元素值为x的结点

    void Divide(LinkList<T> &A,LinkList<T> &B); //将当前链表分解为A，B表

    void DispList( );           //遍历单链表，按序号依次输出各元素

 private:

   Node<T> \*first; //单链表的头指针

};

int main( ){

       LinkList<int> A,B,C;

       int x,i=1;

       while(1)

       {

              cin>>x;

             if(!x)break;

              try{

                     C.Insert(i++,x);

              }

              catch(const char \*ms)

              {

                     cout<<ms<<endl;

              }

       }

       C.DispList();

       C.Divide(A,B);

       C.DispList();

       A.DispList();

       B.DispList();

       return 0;

}

**Input**

**Output**

**主函数利用插入算法创建链表C，再将表C分解为表A，表B，根据C表中的结点的奇偶分别插入到表A或表B的头结点之后。插入后表C置为空表。因为插入到头结点之后，表A和表B中的元素序列与原表中的次序反向。**

**Sample Input**

23 2 42 4 1121 27 40 0

**Sample Output**

The length:7

The elements:

23 2 42 4 1121 27 40

The length:0

The elements:

The length:3

The elements:

27 1121 23

The length:4

The elements:

40 4 42 2

#include <iostream>

using namespace std;

template <class T>

struct Node

{

    T data;

    Node<T> \*next;

};

template <class T>

class LinkList

{

 public:

    LinkList( ); //建立只有头结点的空链表

    ~LinkList();             //析构函数

    void Insert(int i,T x);   //在有序单链表中插入元素x使序列仍有序

    int Length();      //求表长

    void DispList( );     //遍历有序单链表，按序号依次输出各元素

    void Divide(LinkList<T> &A,LinkList<T> &B); //将当前链表分解为A，B表

 private:

   Node<T> \*first; //单链表的头指针

};

template <class T>

LinkList<T>::LinkList()

{

    first=new Node<T>;

    first->next=NULL;

}

template <class T>

LinkList<T>::~LinkList()

{

    Node <T> \*p;

    for(  ; first!=NULL; )

    {   p=first;

        first=first->next;

        delete p;

    }

}

template <class T>

void LinkList<T>::Insert(int i,T x)

{

    int z;

    Node <T> \*p,\*s;

    s=new Node<T>;

    s->data=x;

    for( z=1,p=first ; p->next!=NULL && z<i ; p=p->next ,z++);

    if(z<i)  throw "wrong";

    s->next=p->next;

    p->next=s;

}

template <class T>

int LinkList<T>::Length()

{

    Node<T> \*p;

    int i;

    for(p=first->next ,i=0; p!=NULL ; p=p->next ,i++);

    return i;

}

template <class T>

void LinkList<T>::Divide(LinkList<T> &A,LinkList<T> &B)

{

    Node<T> \*p,\*q,\*s,\*j;

    s=first->next;

    first->next=NULL;

    p=A.first;

    q=B.first;

    for(; s ; )

        if(s->data%2==1)

        {

            j=p->next;

            p->next=s;

            s=s->next;

            p->next->next=j;

        }

        else

        {

            j=q->next;

            q->next=s;

            s=s->next;

            q->next->next=j;

        }

}

template <class T>

void LinkList<T>::DispList()

{

    cout<<"The length:"<<Length()<<endl;

    cout<<"The elements:"<<endl;

    Node<T> \*p;

    for(p=first->next; p!=NULL ; p=p->next)

        cout<<p->data<<" ";

    cout<<endl;

}

int main( ){

       LinkList<int> A,B,C;

       int x,i=1;

       while(1)

       {

              cin>>x;

             if(!x)break;

              try{

                     C.Insert(i++,x);

              }

              catch(const char \*ms)

              {

                     cout<<ms<<endl;

              }

       }

       C.DispList();

       C.Divide(A,B);

       C.DispList();

       A.DispList();

       B.DispList();

       return 0;

}

**1008: 单向循环链表**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 79  Solved: 63  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1008)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1008)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1008)]

**Description**

**单向循环表类的定义及部分函数和main函数代码（请勿改动）如下，请完成Length、Insert、Reverse和DispList算法。**

//单向循环链表类

template <class T>

class LinkList

{

 public:

    LinkList( ); //建立只有头结点的空链表

    ~LinkList(); //析构函数

    int Length();    //求单链表的长度

    void Insert(T x);   //在单链表中的表尾位置插入元素值为x的结点

       void Reverse( );     //逆置链表

    void DispList( );     //遍历单链表，按序号依次输出各元素

 private:

   Node<T> \*first; //单链表的头指针

};

//构造函数

template <class T>

LinkList<T>:: LinkList( ){

       first=new Node<T>;

       first->next=first;

}

//析构函数：释放链表各个结点所占内存单元

template <class T>

LinkList<T>:: ~LinkList( ){

    Node<T> \*s;

       s=first->next;

       while (s!=first){

        first->next=s->next;

              delete s;

              s=first->next;

       }

       delete s;

}

int main( ){

       LinkList<int> sa;

       int x;

       while(1)

       {

              cin>>x;

        if(!x)break;

              sa.Insert(x);

       }

    sa.DispList();

       sa.Reverse();

       sa.DispList();

       return 0;

}

**Input**

**Output**

**Sample Input**

1 2 3 4 5 6 0

**Sample Output**

The length:6

The elements:

1 2 3 4 5 6

The length:6

The elements:

6 5 4 3 2 1

#include <iostream>

using namespace std;

template <class T>

struct Node

{

T data;

Node<T> \*next;

};

template <class T>

class LinkList

{

public:

LinkList( ); //建立只有头结点的空链表

~LinkList(); //析构函数

int Length(); //求单链表的长度

void Insert(T x); //在单链表中的表尾位置插入元素值为x的结点

void Reverse( ); //逆置链表

void DispList( ); //遍历单链表，按序号依次输出各元素

private:

Node<T> \*first; //单链表的头指针

};

template <class T>

LinkList<T>:: LinkList( ){

first=new Node<T>;

first->next=first;

}

template <class T>

LinkList<T>:: ~LinkList( ){

Node<T> \*s;

s=first->next;

while (s!=first){

first->next=s->next;

delete s;

s=first->next;

}

delete s;

}

template <class T>

void LinkList<T>::Insert(T x)

{

Node <T> \*p,\*s;

s=new Node<T>;

s->data=x;

for( p=first ; p->next!=first ; p=p->next );

p->next=s;

s->next=first;

}

template <class T>

int LinkList<T>::Length()

{

Node<T> \*p;

int i;

for(p=first->next ,i=0; p!=first ; p=p->next ,i++);

return i;

}

template <class T>

void LinkList<T>::Reverse( )

{

Node<T> \*p,\*q,\*s,\*t;

t=first->next;

for( p=first->next ; p!=first ; )

{

s=p;

p=p->next;

q=first->next;

first->next=s;

s->next=q;

}

t->next=first;

}

template <class T>

void LinkList<T>::DispList()

{

cout<<"The length:"<<Length()<<endl;

cout<<"The elements:"<<endl;

Node<T> \*p;

for(p=first->next; p!=first ; p=p->next)

cout<<p->data<<" ";

cout<<endl;

}

int main( )

{

LinkList<int> sa;

int x;

while(1)

{

cin>>x;

if(!x)break;

sa.Insert(x);

}

sa.DispList();

sa.Reverse();

sa.DispList();

return 0;

}

**1009: 双向循环链表**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 50  Solved: 42  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1009)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1009)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1009)]

**Description**

**双向循环链表类的定义、构造函数和析构函数实现及主函数如下（不允许改动主函数）。请完成类的其它成员函数实现。**

template <class T>

struct DNode

{

   T data;

   DNode<T> \*prior, \*next;

};

template <class T>

class DLinkList

{

 public:

       DLinkList( ); //建立只有头结点的双向循环链表链表

       ~DLinkList();        //析构函数

       int Length();        //求链表的长度

       void Insert(T x);    //表尾插入x

       void DispListF( );   //正向遍历链表，按照逻辑序号输出各元素

       void DispListR( );   //反向遍历链表，按照反向逻辑序号输出各元素

 private:

   DNode<T> \*first; //链表的头指针

};

 /\*  
\*前置条件：链表不存在  
\*输    入：无  
\*功    能：构建一个只有头结点的双向循环链表  
\*输    出：无  
\*后置条件：构建一个链表  
\*/  
template <class T>  
DLinkList<T>:: DLinkList( ){  
 first=new DNode<T>;   
 first->next=first->prior=first;  
}

/\*  
\*前置条件：链表存在  
\*输    入：无  
\*功    能：释放链表各个结点所占内存单元  
\*输    出：无  
\*后置条件：链表结点被释放  
\*/  
template <class T>  
DLinkList<T>:: ~DLinkList()  
{    
    DNode<T> \*s;  
    s=first->next;  
    while (s!=first){  
        first->next=s->next;  
        delete s;  
       s=first->next;  
    }  
   delete s;  
}

int main()

{

       DLinkList<string> DL;

       string subject;

       int i,n;

       cin>>n;

       for(i=1;i<=n;i++)

       {

              cin>>subject;

              DL.Insert(subject);

       }

       DL.DispListF();

       DL.DispListR();

       return 0;

}

**Input**

**Output**

**Sample Input**

8 zhao qian sun li zhou wu zheng wang

**Sample Output**

The length:8

The elements:

zhao qian sun li zhou wu zheng wang

The length:8

The elements:

wang zheng wu zhou li sun qian zhao

#include<iostream>

#include<string>

using namespace std;

template <class T>

struct DNode

{

T data;

DNode<T> \*prior, \*next;

};

template <class T>

class DLinkList

{

public:

DLinkList( ); //建立只有头结点的双向循环链表链表

~DLinkList(); //析构函数

int Length(); //求链表的长度

void Insert(T x); //表尾插入x

void DispListF( ); //正向遍历链表，按照逻辑序号输出各元素

void DispListR( ); //反向遍历链表，按照反向逻辑序号输出各元素

private:

DNode<T> \*first; //链表的头指针

};

/\*

\*前置条件：链表不存在

\*输 入：无

\*功 能：构建一个只有头结点的双向循环链表

\*输 出：无

\*后置条件：构建一个链表

\*/

template <class T>

DLinkList<T>:: DLinkList( ){

first=new DNode<T>;

first->next=first->prior=first;

}

/\*

\*前置条件：链表存在

\*输 入：无

\*功 能：释放链表各个结点所占内存单元

\*输 出：无

\*后置条件：链表结点被释放

\*/

template <class T>

DLinkList<T>:: ~DLinkList()

{

DNode<T> \*s;

s=first->next;

while (s!=first){

first->next=s->next;

delete s;

s=first->next;

}

delete s;

}

template <class T>

void DLinkList<T>::DispListF()

{

cout<<"The length:"<<Length()<<endl<<"The elements:"<<endl;

DNode<T> \*p;

for(p=first->next ; p!=first ; p=p->next)

cout<<p->data<<" ";

cout<<endl;

}

template <class T>

void DLinkList<T>::DispListR()

{

cout<<"The length:"<<Length()<<endl<<"The elements:"<<endl;

DNode<T> \*p;

for(p=first->prior ; p!=first ; p=p->prior)

cout<<p->data<<" ";

cout<<endl;

}

template <class T>

int DLinkList<T>::Length()

{

int i;

DNode<T> \*p;

p=first->next;

for(i=0 ; p!=first ; p=p->next,i++);

return i;

}

template <class T>

void DLinkList<T>::Insert(T x)

{

DNode<T> \*p,\*s;

s=new DNode<T>;

s->data=x;

p=first;

for( ; p->next!=first ; p=p->next );

s->prior=p;

first->prior=s;

s->next=p->next;

p->next=s;

}

int main()

{

DLinkList<string> DL;

string subject;

int i,n;

cin>>n;

for(i=1 ; i<=n ; i++)

{

cin>>subject;

DL.Insert(subject);

}

DL.DispListF();

DL.DispListR();

return 0;

}

**1010: 约瑟夫（Joseff）问题求解（1）**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 37  Solved: 26  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1010)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1010)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1010)]

**Description**

**约瑟夫（Joseff）问题求解。n个人围成一圈，编号依次为1,2，。。。。n，从第一个人开始报数，m号出圈，再从下一个开始报数，。。。，m号出圈，直至所有人出圈。求出圈的次序。要求利用数组实现存储。**

**Input**

**输入数据依次为n和m**

**Output**

**Sample Input**

10 5

**Sample Output**

5 10 6 2 9 8 1 4 7 3

#include<iostream>

using namespace std;

int main()

{

int i,n,m,z,j,a[10000];

cin>>n>>m;

for(i=1 ; i<=n ; i++)

a[i]=i;

for( j=1,i=1 ; n ; )

{

if(j==m)

{

cout<<a[i]<<" ";

for( z=i ; z<n ; z++)

a[z]=a[z+1];

if(i==n) i=1;

j=1;

n--;

}

else {

j++;

if(i==n) i=1;

else i++;

}

}

return 0;

}

**1011: 约瑟夫（Joseff）问题求解（2）**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 28  Solved: 20  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1011)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1011)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1011)]

**Description**

**约瑟夫（Joseff）问题求解。n个人围成一圈，编号依次为1,2，。。。。n，从第一个人开始报数，m号出圈，再从下一个开始报数，。。。，m号出圈，直至所有人出圈。求出圈的次序。**

**要求采用仅设尾指针的单向循环链表实现。已知类的定义、部分成员函数及主函数代码如下（勿改动）。**

//结点结构  
template <class T>  
struct Node  
{  
  T data;  
  Node<T> \*next;    
};  
//单向循环链表类  
template <class T>  
class LinkList  
{  
  public:  
    LinkList( );  //建立没有头结点的空链表  
    ~LinkList();  //析构函数  
 void RInsert(T x);   //在链表表尾插入元素x  
 void Create(int n);  //利用RInsert创建仅有尾指针的单向循环链表（不设置头结点）  
 void Joseff(int m);   // 约瑟夫问题求解  
    void DispList( );     //遍历单链表，按序号依次输出各元素  
 private:  
   Node<T> \*rear;  //尾指针  
};

//创建空的循环表  
template <class T>  
LinkList<T>:: LinkList( ){  
 rear=NULL;  
}

//析构函数：释放链表结点所占内存单元  
template <class T>    
LinkList<T>:: ~LinkList( ){     
    Node<T> \*s;  
 if(rear) //链表非空析构  
 {  
  s=rear->next;  
  while (s!=rear){  
   rear->next=s->next;  
   delete s;  
   s=rear->next;  
  }  
  delete s;  
 }  
}  
//创建n个数据的链表  
template <class T>    
void LinkList<T>::Create(int n){     
    int i;  
 for(i=1;i<=n;i++)  
  RInsert(i);  
}  
/\*  
\*前置条件：单链表存在  
\*输    入：无  
\*功    能：单链表遍历  
\*输    出：输出所有元素  
\*后置条件：单链表不变  
\*/  
template <class T>   
void LinkList<T>::DispList( )  
{  
 Node<T> \*p;  
  if(rear)  //链表非空输出数据  
 {  
  for(p=rear->next;p!=rear;p=p->next)  
   cout<<p->data<<" ";  
  cout<<p->data<<" ";  
 }  
 cout<<endl;  
}  
int main( ){  
 LinkList<int> sa;  
 int m,n;  
 cin>>n>>m;  
 sa.Create(n);  //创建n个人（编号1~n）的链表  
    sa.DispList(); // 遍历链表，输出编号  
 sa.Joseff(m);  // 输出出圈人的编号次序  
 return 0;  
}

**Input**

输入次序为n，m

**Output**

**Sample Input**

10 5

**Sample Output**

1 2 3 4 5 6 7 8 9 10

5 10 6 2 9 8 1 4 7 3

#include<iostream>

using namespace std;

template <class T>

struct Node

{

T data;

Node<T> \*next;

};

//单向循环链表类

template <class T>

class LinkList

{

public:

LinkList( ); //建立没有头结点的空链表

~LinkList(); //析构函数

void RInsert(T x); //在链表表尾插入元素x

void Create(int n); //利用RInsert创建仅有尾指针的单向循环链表（不设置头结点）

void Joseff(int m); // 约瑟夫问题求解

void DispList( ); //遍历单链表，按序号依次输出各元素

private:

Node<T> \*rear; //尾指针

};

//创建空的循环表

template <class T>

LinkList<T>:: LinkList( )

{

rear=NULL;

}

//析构函数：释放链表结点所占内存单元

template <class T>

LinkList<T>:: ~LinkList( ){

Node<T> \*s;

if(rear) //链表非空析构

{

s=rear->next;

while (s!=rear){

rear->next=s->next;

delete s;

s=rear->next;

}

delete s;

}

}

//创建n个数据的链表

template <class T>

void LinkList<T>::Create(int n)

{

int i;

for(i=1;i<=n;i++)

RInsert(i);

}

template <class T>

void LinkList<T>::RInsert(T x)

{

Node<T> \*s;

s=new Node<T>;

s->data=x;

if(rear)

{

s->next=rear->next;

rear->next=s;

rear=s;

}

else

{

s->next=s;

rear=s;

}

}

template <class T>

void LinkList<T>::Joseff(int m)

{

Node<T> \*s;

int i=1;

for(s=rear; ; )

{

if(i==m)

{

i=1;

if(rear==rear->next) //如果只剩一个表了就输出并跳出循环

{

cout<<rear->data<<endl;

break;

}

if(s->next==rear) rear=s;//如果点到名的是最后一位，删除后让尾指针向前移

cout<<s->next->data<<" ";

s->next=s->next->next;

}

else

{

i++;

s=s->next; //或者s->next=s=s->next->next;

}

}

}/\*

\*前置条件：单链表存在

\*输 入：无

\*功 能：单链表遍历

\*输 出：输出所有元素

\*后置条件：单链表不变

\*/

template <class T>

void LinkList<T>::DispList( )

{

Node<T> \*p;

if(rear) //链表非空输出数据

{

for(p=rear->next;p!=rear;p=p->next)

cout<<p->data<<" ";

cout<<p->data<<" ";

}

cout<<endl;

}

int main( ){

LinkList<int> sa;

int m,n;

cin>>n>>m;

sa.Create(n); //创建n个人（编号1~n）的链表

sa.DispList(); // 遍历链表，输出编号

sa.Joseff(m); // 输出出圈人的编号次序

return 0;

}