## 1032: 散列表（1）

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 44  Solved: 22  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1032)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1032)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1032)]

## Description

**已知Hash表的表长MaxSize为11，Hash函数为HashFunc（k）=k%11，冲突处理方法为线性探测法，部分代码如下，勿改动。请补充完成成员函数HashSearch，该函数的功能是动态查找k，若查找失败，则插入k，并返回查找失败所需的比较次数，若查找成功，返回查找k所需的比较次数，若表满，则抛出异常“Overflow”**

const int MaxSize=11; //hash表的表长，即教材中的m

class HashList  
{  
private:  
 int ht[MaxSize]; // hashtable  
public:  
 int HashFunc(int k);  //hash function  
 HashList();     //consturctor  
 void Display();      //display   
 int HashSearch(int k);  //dynamic search k  
 double HashASL();   //search ASL  
};

//hash function  
int HashList::HashFunc(int k)  
{  
 return k%11;   //hash函数，假设为除余法，余数为11  
}

//constructor:initialize an empty hashlist  
HashList::HashList()  
{  
 int i;  
 for(i=0;i<MaxSize;i++)  
  ht[i]=-1;  //-1 is empty  
}  
void HashList::Display()  
{  
 int i;  
 for(i=0;i<MaxSize;i++)  
  cout<<ht[i]<<" ";  
 cout<<endl;  
}

double  HashList::HashASL()   
{  
 double ASL=0;  
 int i,n=0;  
 for(i=0;i<MaxSize;i++)  
  if(ht[i]!=-1)  
  {  
   ASL+=HashSearch(ht[i]);  
   cout<<ht[i]<<" "<<HashSearch(ht[i])<<endl;  
   n++;  
  }  
 return ASL/n;  
}

int main()  
{  
    HashList HL;  
// HL.Display();  
 while(1)  
 {  
  int k;  
  cin>>k;  
  if(!k) break;  
  try{  
   HL.HashSearch(k);   
  }  
  catch(const char \*ms)  
  {  
   cout<<ms<<endl;  
  }  
 }  
 HL.Display();    
 cout<<HL.HashASL();  
 return 0;  
}

**Input**

输入数据以0结束，依次实现动态查找。

**Output**

输出数据 （各行数据依次为）：

hash表中的数据的输出；

各个元素的值及其查找次数；

ASL值

**Sample Input**

47 7 29 11 16 92 22 8 3 29 0

**Sample Output**

11 22 -1 47 92 16 3 7 29 8 -1

11 1

22 2

47 1

92 1

16 1

3 4

7 1

29 2

8 2

ASL=1.66667

#include<iostream>

using namespace std;

const int MaxSize=11; //hash表的表长，即教材中的m

class HashList

{

private:

int ht[MaxSize]; // hashtable

public:

int HashFunc(int k); //hash function

HashList(); //consturctor

void Display(); //display

int HashSearch(int k); //dynamic search k

double HashASL(); //search ASL

};

//hash function

int HashList::HashFunc(int k)

{

return k%11; //hash函数，假设为除余法，余数为11

}

//constructor:initialize an empty hashlist

HashList::HashList()

{

int i;

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

ht[i]=-1; //-1 is empty

}

void HashList::Display()

{

int i;

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

cout<<ht[i]<<" ";

cout<<endl;

}

double HashList::HashASL()

{

double ASL=0;

int i,n=0;

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

if(ht[i]!=-1)

{

ASL+=HashSearch(ht[i]);

cout<<ht[i]<<" "<<HashSearch(ht[i])<<endl;

n++;

}

return ASL/n;

}

int HashList::HashSearch(int k)

{

int i,count=0;

for( i=HashFunc(k) ; count <= MaxSize ; i=(i+1)%MaxSize )

{

count++;

if( ht[i] == k ) break;

if( ht[i] == -1) {

ht[i] = k;

break;

}

if( count == MaxSize ) throw "Overflow";

}

return count;

}

int main()

{

HashList HL;

// HL.Display();

while(1)

{

int k;

cin>>k;

if(!k) break;

try{

HL.HashSearch(k);

}

catch(const char \*ms)

{

cout<<ms<<endl;

}

}

HL.Display();

cout<<"ASL="<<HL.HashASL()<<endl;

return 0;

}

## 1033: 散列表（2）

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 30  Solved: 24  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1033)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1033)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1033)]

## Description

已知Hash表的表长MaxSize为11，Hash函数为HashFunc（k）=k%11，冲突处理方法为链地址法，部分代码如下，勿改动。请补充完成成员函数HashSearch，该函数的功能是动态查找k，若查找失败，则插入k，并返回查找失败所需的比较次数，若查找成功，返回查找k所需的比较次数。

#include<iostream>

using namespace std;

const int MaxSize=11;    //maxsize

struct Node

{

int data;

    Node \*next;

};

class LinkHash

{

public:

LinkHash();  //initialize an empty list

int HashFunc(int k);  //hash function

int HashSearch(int k); //dynamic search k

void Display();

private:

   Node \*ht[MaxSize];  //ht数组用来保留各个链表的头指针

};

//hash function

int LinkHash::HashFunc(int k)

{

return k%11;   //hash函数，假设为除余法，余数为11

}

//constructor:initialize an empty hashlist

LinkHash::LinkHash()

{

int i;

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

ht[i]=NULL;  //NULL is empty

}

void LinkHash::Display()

{

int i;

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

{

cout<<"Hash address:"<<i<<",value:";

Node \*p;

for(p=ht[i];p;p=p->next)

cout<<p->data<<" ";

cout<<endl;

}

}

int main()

{

LinkHash LS;

  int k;

while(1)

{

cin>>k;

if(!k) break;

try{

LS.HashSearch(k);

// LS.Display();

}

catch(const char \*ms)

{

cout<<ms<<endl;

}

}

LS.Display();

return 0;

}

**Input**

**Output**

**Sample Input**

47 7 29 11 16 92 22 8 3 29 0

**Sample Output**

Hash address:0,value:22 11

Hash address:1,value:

Hash address:2,value:

Hash address:3,value:3 47

Hash address:4,value:92

Hash address:5,value:16

Hash address:6,value:

Hash address:7,value:29 7

Hash address:8,value:8

Hash address:9,value:

Hash address:10,value:

#include<iostream>

using namespace std;

const int MaxSize=11; //maxsize

struct Node

{

int data;

Node \*next;

};

class LinkHash

{

public:

LinkHash(); //initialize an empty list

int HashFunc(int k); //hash function

int HashSearch(int k); //dynamic search k

void Display();

private:

Node \*ht[MaxSize]; //ht数组用来保留各个链表的头指针

};

//hash function

int LinkHash::HashFunc(int k)

{

return k%11; //hash函数，假设为除余法，余数为11

}

//constructor:initialize an empty hashlist

LinkHash::LinkHash()

{

int i;

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

ht[i]=NULL; //NULL is empty

}

void LinkHash::Display()

{

int i;

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

{

cout<<"Hash address:"<<i<<",value:";

Node \*p;

for(p=ht[i];p;p=p->next)

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

cout<<endl;

}

}

int LinkHash::HashSearch(int k)

{

int count=0;

Node \*p,\*s;

for( p = ht[HashFunc(k)] ; p ; p=p->next )

{

count++;

if( p -> data == k ) break;

}

if( !p ) {

s=new Node;

s -> data =k;

p = ht[HashFunc(k)];

ht[HashFunc(k)] = s;

s -> next = p;

}

return count;

}

int main()

{

LinkHash LS;

int k;

while(1)

{

cin>>k;

if(!k) break;

try{

LS.HashSearch(k);

// LS.Display();

}

catch(const char \*ms)

{

cout<<ms<<endl;

}

}

LS.Display();

return 0;

}

**1034: 直接插入排序**

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=1034)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1034)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1034)]

**Description**

**已知直接插入排序的部分代码如下，勿改动，请补充实现插入排序函数。**

#include<iostream>  
using namespace std;  
const int MaxSize=100;  
class List  
{  
private:  
 int r[MaxSize+1];  
 int n;  
public:  
    List(){n=0;} //empty list  
 void InsertR(int k)  //表尾插入  
 {  r[++n]=k;}  
 void Display();      //display  
 void InsertSort();   //InsertSort  
};  
void List::Display()  
{  
 for(int i=1;i<=n;i++)  
       cout<<r[i]<<" ";     
 cout<<"\n";  
}  
int main()  
{  
 List L;  
 while(1)  
 {  
  int k;  
  cin>>k;  
  if(!k) break;  
        L.InsertR(k);  
 }  
 L.Display();  
 L.InsertSort();  
 L.Display();  
 return 0;  
}

**Input**

**Output**

**Sample Input**

12 21 32 2 4 24 21 432 23 9 0

**Sample Output**

12 21 32 2 4 24 21 432 23 9

2 4 9 12 21 21 23 24 32 432

#include<iostream>

using namespace std;

const int MaxSize=100;

class List

{

private:

int r[MaxSize+1];

int n;

public:

List(){n=0;} //empty list

void InsertR(int k) //表尾插入

{ r[++n]=k;}

void Display(); //display

void InsertSort(); //SelectSort

};

void List::InsertSort()

{

int i,j;

for( i = 2 ; i <= n ; i++ )

{

r[0]=r[i];

for( j = i-1 ; r[0] < r[j] ; j-- )

r[j+1] = r[j];

r[j+1] = r[0];

}

}

void List::Display()

{

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

cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

L.InsertR(k);

}

L.Display();

L.InsertSort();

L.Display();

return 0;

}

**1035: 希尔排序**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 48  Solved: 37  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1035)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1035)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1035)]

**Description**

**已知希尔排序的部分代码如下，勿改动，请补充实现希尔排序函数。**

#include<iostream>  
using namespace std;  
const int MaxSize=100;  
class List  
{  
private:  
 int r[MaxSize+1];  
 int n;  
public:  
    List(){n=0;} //empty list  
 void InsertR(int k)  //表尾插入  
 {  r[++n]=k;}  
 void Display();      //display  
 void ShellSort();   //ShellSort  
};  
void List::Display()  
{  
 for(int i=1;i<=n;i++)  
       cout<<r[i]<<" ";     
 cout<<"\n";  
}  
int main()  
{  
 List L;  
 while(1)  
 {  
  int k;  
  cin>>k;  
  if(!k) break;  
        L.InsertR(k);  
 }  
 L.Display();  
 L.ShellSort();  
 L.Display();  
 return 0;  
}

**Input**

**Output**

**Sample Input**

12 21 32 2 4 24 21 432 23 9 0

**Sample Output**

12 21 32 2 4 24 21 432 23 9

2 4 9 12 21 21 23 24 32 432

#include<iostream>

using namespace std;

const int MaxSize=100;

class List

{

private:

int r[MaxSize+1];

int n;

public:

List(){n=0;} //empty list

void InsertR(int k) //表尾插入

{ r[++n]=k;}

void Display(); //display

void ShellSort(); //ShellSort

};

void List::ShellSort()

{

int d,i,j;

for( d = n/2 ; d >= 1 ; d/=2 )

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

{

r[0] = r[i];

for( j = i-d ; j>0 && r[0] < r[j] ; j-=d )

r[j+d] = r[j];

r[j+d] = r[0];

}

}

void List::Display()

{

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

cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

L.InsertR(k);

}

L.Display();

L.ShellSort();

L.Display();

return 0;

}

## 1036: 起泡排序

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 36  Solved: 30  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1036)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1036)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1036)]

## Description

**已知起泡排序的部分代码如下，勿改动，请补充实现起泡排序函数。**

#include<iostream>  
using namespace std;  
const int MaxSize=100;  
class List  
{  
private:  
 int r[MaxSize+1];  
 int n;  
public:  
    List(){n=0;} //empty list  
 void InsertR(int k)  //表尾插入  
 {  r[++n]=k;}  
 void Display();      //display  
 void BubbleSort();   //BubbleSort  
};  
void List::Display()  
{  
 for(int i=1;i<=n;i++)  
       cout<<r[i]<<" ";     
 cout<<"\n";  
}  
int main()  
{  
 List L;  
 while(1)  
 {  
  int k;  
  cin>>k;  
  if(!k) break;  
        L.InsertR(k);  
 }  
 L.Display();  
 L.BubbleSort();  
 L.Display();  
 return 0;  
}

**Input**

**Output**

**Sample Input**

12 21 32 2 4 24 21 432 23 9 0

**Sample Output**

12 21 32 2 4 24 21 432 23 9

2 4 9 12 21 21 23 24 32 432

#include<iostream>

using namespace std;

const int MaxSize=100;

class List

{

private:

int r[MaxSize+1];

int n;

public:

List(){n=0;} //empty list

void InsertR(int k) //表尾插入

{ r[++n]=k;}

void Display(); //display

void BubbleSort(); //BubbleSort

};

void List::BubbleSort()

{

int i,exchange=n,bound,temp;

while( exchange )

{

bound=exchange;

exchange=0;

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

if( r[i] > r[i+1] ){

temp=r[i];

r[i]=r[i+1];

r[i+1]=temp;

exchange=i;

}

}

}

void List::Display()

{

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

cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

L.InsertR(k);

}

L.Display();

L.BubbleSort();

L.Display();

return 0;

}

## 1037: 快速排序（1）

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 39  Solved: 27  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1037)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1037)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1037)]

## Description

**已知快速排序的部分代码如下，勿改动，请补充实现快速排序函数：void QuickSort(int first,int end);   //quickSort**

#include<iostream>

using namespace std;

const int MaxSize=100;

class List

{

private:

int r[MaxSize+1];

int n;

public:

    List(){n=0;} //empty list

void InsertR(int k)  //表尾插入

{  r[++n]=k;}

void Display();      //display

void QuickSort(int first,int end);   //quickSort

void QuickSort()

{

QuickSort(1,n);

}

};

void List::Display()

{

for(int i=1;i<=n;i++)

       cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

        L.InsertR(k);

}

L.Display();

L.QuickSort();

L.Display();

return 0;

}

**Input**

**Output**

**Sample Input**

12 21 32 2 4 24 21 432 23 9 0

**Sample Output**

12 21 32 2 4 24 21 432 23 9

2 4 9 12 21 21 23 24 32 432

#include<iostream>

using namespace std;

const int MaxSize=100;

class List

{

private:

int r[MaxSize+1];

int n;

public:

List(){n=0;} //empty list

void InsertR(int k) //表尾插入

{ r[++n]=k;}

void Display(); //display

void QuickSort(int first,int end); //quickSort

void QuickSort()

{

QuickSort(1,n);

}

};

void List::QuickSort(int first, int end)

{

r[0]=r[first];

if( first < end )

{

int i=first, j=end ;

while( i < j )

{

while( r[0] < r[j] && i<j )

j--;

if(i < j)

{

r[i] = r[j]; //将s[j]填到s[i]中，s[j]就形成了一个新的坑

i++;

}

while( r[0] > r[i] && i<j )

i++;

if(i < j)

{

r[j] = r[i]; //将s[i]填到s[j]中，s[i]就形成了一个新的坑

j--;

}

}

r[i] = r[0];

QuickSort( first, i-1 );

QuickSort( i+1 , end);

}

}

void List::Display()

{

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

cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

L.InsertR(k);

}

L.Display();

L.QuickSort();

L.Display();

return 0;

}

## 1038: 快速排序（2）

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 16  Solved: 15  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1038)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1038)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1038)]

## Description

**已知快速排序的部分代码如下，勿改动，请利用栈实现快速排序非递归函数：void QuickSort();   //quickSort**

#include<iostream>

using namespace std;

const int MaxSize=100;

class List

{

private:

int r[MaxSize+1];

int n;

public:

    List(){n=0;} //empty list

void InsertR(int k)  //表尾插入

{  r[++n]=k;}

void Display();      //display

void QuickSort();   //quickSort

};

void List::Display()

{

for(int i=1;i<=n;i++)

       cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

        L.InsertR(k);

}

L.Display();

L.QuickSort();

L.Display();

return 0;

}

**Input**

**Output**

**Sample Input**

12 21 32 2 4 24 21 432 23 9 0

**Sample Output**

12 21 32 2 4 24 21 432 23 9

2 4 9 12 21 21 23 24 32 432

#include<iostream>

using namespace std;

const int MaxSize=100;

const int StackSize=100;

class SeqStack

{

private:

int n;

int data[StackSize];

public:

SeqStack();

~SeqStack(){}

void Push(int x);

int Pop();

int GetTop();

bool Empty();

};

SeqStack::SeqStack()

{

n=-1;

}

void SeqStack::Push( int x)

{

data[++n]=x;

}

int SeqStack::Pop()

{

if( !Empty() )

{

int x;

x=data[n--];

return x;

}

else throw "Downflow";

}

int SeqStack::GetTop()

{

if( !Empty() )

return data[n];

else throw "Downflow";

}

bool SeqStack::Empty()

{

if( n == -1 ) return true;

else return false;

}

class List

{

private:

int r[MaxSize+1];

int n;

public:

List(){n=0;} //empty list

void InsertR(int k) //表尾插入

{ r[++n]=k;}

void Display(); //display

void QuickSort(); //quickSort

};

void List::QuickSort()

{

SeqStack s;

s.Push(1);

s.Push(n);

int first,end,i,j;

while( !s.Empty() )

{

end=j=s.Pop();

first=i=s.Pop();

if( i < j)

{

r[0] = r[i];

while( i < j )

{

for( ; i < j && r[0] < r[j] ; j-- );

if( i < j ) r[i++] = r[j];

for( ; i < j && r[0] > r[i] ; i++ );

if( i < j ) r[j--] = r[i];

}

r[i] = r[0];

s.Push( first );

s.Push( i-1 );

s.Push( j+1 );

s.Push( end );

}

}

}

void List::Display()

{

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

cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

L.InsertR(k);

}

L.Display();

L.QuickSort();

L.Display();

return 0;

}

## 1039: 简单选择排序

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 26  Solved: 21  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1039)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1039)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1039)]

## Description

**已知简单选择排序的部分代码如下，勿改动，请补充实现简单选择排序函数。**

#include<iostream>  
using namespace std;  
const int MaxSize=100;  
class List  
{  
private:  
 int r[MaxSize+1];  
 int n;  
public:  
    List(){n=0;} //empty list  
 void InsertR(int k)  //表尾插入  
 {  r[++n]=k;}  
 void Display();      //display  
 void SelectSort();   //SelectSort  
};  
void List::Display()  
{  
 for(int i=1;i<=n;i++)  
       cout<<r[i]<<" ";     
 cout<<"\n";  
}  
int main()  
{  
 List L;  
 while(1)  
 {  
  int k;  
  cin>>k;  
  if(!k) break;  
        L.InsertR(k);  
 }  
 L.Display();  
 L.SelectSort();  
 L.Display();  
 return 0;  
}

**Input**

**Output**

**Sample Input**

12 21 32 2 4 24 21 432 23 9 0

**Sample Output**

12 21 32 2 4 24 21 432 23 9

2 4 9 12 21 21 23 24 32 432

#include<iostream>

using namespace std;

const int MaxSize=100;

class List

{

private:

int r[MaxSize+1];

int n;

public:

List(){n=0;} //empty list

void InsertR(int k) //表尾插入

{ r[++n]=k;}

void Display(); //display

void SelectSort(); //SelectSort

};

void List::SelectSort()

{

int i,j,index,temp;

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

{

index=i;

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

if( r[j] < r[index] ) index=j;

if( index != i ) {

temp=r[i];

r[i]=r[index];

r[index]=temp;

}

}

}

void List::Display()

{

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

cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

L.InsertR(k);

}

L.Display();

L.SelectSort();

L.Display();

return 0;

}

## 1040: 堆排序

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 12  Solved: 12  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1040)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1040)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1040)]

## Description

**已知堆排序的部分代码如下，勿改动，请补充实现堆排序函数void HeapSort(int n);。**

#include<iostream>  
using namespace std;  
const int MaxSize=100;  
class List  
{  
private:  
 int r[MaxSize+1];  
 int n;  
public:  
    List(){n=0;} //empty list  
 void InsertR(int k)  //表尾插入  
 {  r[++n]=k;}  
 void Display();      //display  
 void HeapSort() { HeapSort(n); }  //HeapSort

void HeapSort(int n);  
};  
void List::Display()  
{  
 for(int i=1;i<=n;i++)  
       cout<<r[i]<<" ";     
 cout<<"\n";  
}  
int main()  
{  
 List L;  
 while(1)  
 {  
  int k;  
  cin>>k;  
  if(!k) break;  
        L.InsertR(k);  
 }  
 L.Display();  
 L.HeapSort();  
 L.Display();  
 return 0;  
}

**Input**

**Output**

**Sample Input**

12 21 32 2 4 24 21 432 23 9 0

**Sample Output**

12 21 32 2 4 24 21 432 23 9

2 4 9 12 21 21 23 24 32 432

#include<iostream>

using namespace std;

const int MaxSize=100;

class List

{

private:

int r[MaxSize+1];

int n;

public:

List(){n=0;} //empty list

void InsertR(int k) //表尾插入

{ r[++n]=k;}

void Display(); //display

void h(int n,int m);

void HeapSort() { HeapSort(n); } //HeapSort

void HeapSort(int n);

};

void List::HeapSort(int n)

{

int i;

for( i = n/2 ; i >= 1 ; i--) //建立初始堆

h( i, n);

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

{

r[0] = r[1];

r[1] = r[n-i+1];

r[n-i+1] = r[0]; //堆顶和末尾交换值

h( 1 , n-i ); //重新调整堆

}

}

void List::h(int first, int end) //第一次局部排堆

{

int i=first,j=i\*2;

while( j <= end )

{

if( j < end && r[j] < r[j+1] )

j++;

if( r[i] < r[j] )

{

r[0]=r[i];

r[i]=r[j];

r[j]=r[0];

i=j;

j=i\*2;

}

else break;

}

}

void List::Display()

{

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

cout<<r[i]<<" ";

cout<<"\n";

}

int main()

{

List L;

while(1)

{

int k;

cin>>k;

if(!k) break;

L.InsertR(k);

}

L.Display();

L.HeapSort();

L.Display();

return 0;

}

## 1042: 散列表（3）

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 12  Solved: 10  
[[Submit](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/submitpage.php?id=1042)][[Status](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/problemstatus.php?id=1042)][[Web Board](http://221.203.21.203:8001/rwt/USTL/http/GV6T6N3UFZ5UGLSSGM3B/OnlineJudge/bbs.php?pid=1042)]

## Description

**已知Hash表的表长MaxSize为11，Hash函数为HashFunc（k）=k%11，冲突处理方法为线性探测法，部分代码如下，勿改动。请补充完成成员函数HashSearch，和函数HashASL，其中函数HashSearch的功能是动态查找k，若查找失败，返回查找失败所需的比较次数，若查找成功，返回查找k所需的比较次数。HashASL的功能是计算等概论查找成功时的ASL值。**

#include<iostream>

using namespace std;

const int MaxSize=11;    //maxsize

struct Node

{

int data;

    Node \*next;

};

class LinkHash

{

public:

LinkHash();  //initialize an empty list

int HashFunc(int k);  //hash function

int HashSearch(int k); //dynamic search k

void Display();

    double HashASL();

private:

   Node \*ht[MaxSize];  //ht数组用来保留各个链表的头指针

};

//hash function

int LinkHash::HashFunc(int k)

{

return k%11;   //hash函数，假设为除余法，余数为11

}

//constructor:initialize an empty hashlist

LinkHash::LinkHash()

{

int i;

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

ht[i]=NULL;  //NULL is empty

}

void LinkHash::Display()

{

int i;

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

{

cout<<"Hash address:"<<i<<",value:";

Node \*p;

for(p=ht[i];p;p=p->next)

cout<<p->data<<" ";

cout<<endl;

}

}

int main()

{

LinkHash LS;

  int k;

while(1)

{

cin>>k;

if(!k) break;

try{

LS.HashSearch(k);

// LS.Display();

}

catch(const char \*ms)

{

cout<<ms<<endl;

}

}

LS.Display();

cout<<"ASL="<<LS.HashASL();

return 0;

}

**Input**

**Output**

**Sample Input**

47 7 29 11 16 92 22 8 3 29 0

**Sample Output**

Hash address:0,value:22 11

Hash address:1,value:

Hash address:2,value:

Hash address:3,value:3 47

Hash address:4,value:92

Hash address:5,value:16

Hash address:6,value:

Hash address:7,value:29 7

Hash address:8,value:8

Hash address:9,value:

Hash address:10,value:

ASL=1.33333

#include<iostream>

using namespace std;

const int MaxSize=11; //maxsize

struct Node

{

int data;

Node \*next;

};

class LinkHash

{

public:

LinkHash(); //initialize an empty list

int HashFunc(int k); //hash function

int HashSearch(int k); //dynamic search k

void Display();

double HashASL();

private:

Node \*ht[MaxSize]; //ht数组用来保留各个链表的头指针

};

//hash function

int LinkHash::HashFunc(int k)

{

return k%11; //hash函数，假设为除余法，余数为11

}

//constructor:initialize an empty hashlist

LinkHash::LinkHash()

{

int i;

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

ht[i]=NULL; //NULL is empty

}

void LinkHash::Display()

{

int i;

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

{

cout<<"Hash address:"<<i<<",value:";

Node \*p;

for(p=ht[i];p;p=p->next)

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

cout<<endl;

}

}

int LinkHash::HashSearch(int k)

{

int count=0;

Node \*p,\*s;

for( p = ht[HashFunc(k)] ; p ; p = p -> next )

{

count++;

if( p -> data == k ) break;

}

if( !p ){

s=new Node;

s -> data = k ;

p = ht[HashFunc(k)];

ht[HashFunc(k)] = s;

s->next = p;

}

return count;

}

double LinkHash::HashASL()

{

int i,n,count=0,num=0;

Node \*p;

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

{

n=0;

for( p = ht[i] ; p ; p= p -> next)

{

num++;

n++;

}

count+=((1+n)\*n/2);

}

return count\*1.0/num;

}

int main()

{

LinkHash LS;

int k;

while(1)

{

cin>>k;

if(!k) break;

try{

LS.HashSearch(k);

// LS.Display();

}

catch(const char \*ms)

{

cout<<ms<<endl;

}

}

LS.Display();

cout<<"ASL="<<LS.HashASL();

return 0;

}