

A C M



秘

籍

目录：

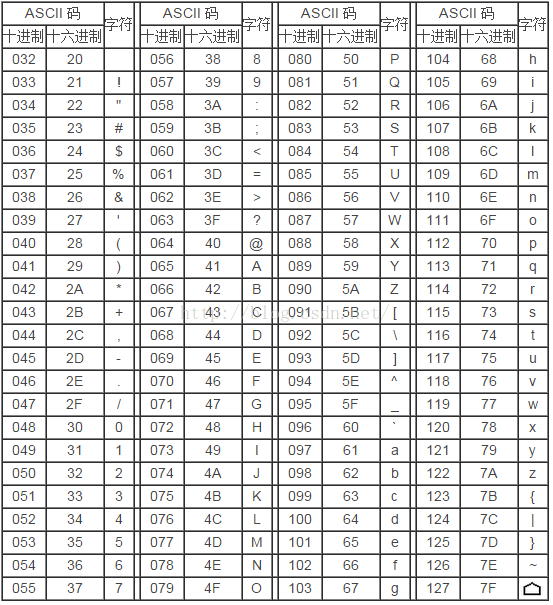
1：ascii ------------------------------------------------2

2：函数头文件--------------------------------------------3

3：ACM竞赛中提交题目时常见的一些结果--------------------4

4：例题--------------------------------------------------5

ASCII码:



函数头文件：

#define \_CRT\_SBCURE\_NO\_DEPRECATE

#include <set>

#include <cmath>

#include <queue>

#include <stack>

#include <vector>

#include <string>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <functional>

using namespace std;

const int maxn = 110;

const int INF = 0x3f3f3f3f;123456789101112131415161718

**ACM竞赛中提交题目时常见的一些结果:**

**1.Output Limit Exceed**是超过输出限制（OLE）错误;  
提示你的程序产生了过多的输出信息，一般是由于死循环造成的。  
你的程序发生上述错误的主要原因是循环语句while(scanf("%d",&n))产生的。  
也有可能是输入字符时未使用getchar()缓冲

**2.memory limit exceed**所用内存超出限制;  
  
**3.Presentation Error**(1).格式错 ( Presentation Error ): 程序没按规定的格式输出答案,指的就是你的输出和规范输出只有空格回车的差异 复杂来说, 就是你的输出和规范输出不分歧 但是把两者中的空格和换行都删掉后却是...  
(2).表达格式错误  
  
**4.Runtime error**

就是在运行期间出现的错误,运行时错误不同于炸弹或系统垮掉,运行时错误一般不影响操作系统运行.  
格式：RUNTIME ERROR SSS AT III：MMM ，SSS是运行错误代码，III是错误发生的程序段，MMM是错误地址。  
出现问题时可以查询含义并找到问题的原因。

runtime error （运行时错误）就是程序运行到一半，程序就崩溃了。

比如说：

①除以零

②数组越界：int a[3]; a[10000000]=10;

③指针越界：int \* p; p=(int \*)malloc(5 \* sizeof(int)); \*(p+1000000)=10;

④使用已经释放的空间：int \* p; p=(int \*)malloc(5 \* sizeof(int));free(p); \*p=10;

⑤数组开得太大，超出了栈的范围，造成栈溢出：int a[100000000];

**5.Time Limit Exceeded**一般是由于程序运行时间过长导致超过限定时间， 可以通过调整算法，优化代码等措施解决  
  
**6.Compilation Error**编译错误  
  
**7.Wrong Answer**答案错误  
  
**8.Accepted**通过

**例题目录：**

**1：打印图形（放大的X）--------------------------------------------------------------7**

**2：背包(采药)-------------------------------------------------------------------------11**

**3：首字母转换为大写-----------------------------------------------------------------16**

**4：生搓大数加法（C）---------------------------------------------------------------18**

**5：全排列-----------------------------------------------------------------------------21**

**6：杨辉三角--------------------------------------------------------------------------23**

**7：开关-------------------------------------------------------------------------------25**

**8：集合相减--------------------------------------------------------------------------27**

**9：集合相加--------------------------------------------------------------------------30**

**10：A==B（高精）------------------------------------------------------------------33**

**11：数字根---------------------------------------------------------------------------36**

**12：抠“ 5“------------------------------------------------------------------------38**

**13：斐波那契数列（附打表代码）----------------------------------------------------41**

请你编程画一个放大的’X’。   
如3\*3的’X’应如下所示：

X X  
 X  
X X  
  
5\*5的’X’如下所示：  
X X  
 X X  
 X  
 X X  
X X

**Input**

输入数据第一行是一个整数T，表示有T组测试数据；   
接下来有T行，每行有一个正奇数n（3 <= n <= 79），表示放大的规格。

**Output**

对于每一个n打印一个规格为n \* n放大的’X’；每组输出后面空一行。

**Sample Input**

2

3

5

**Sample Output**

X X

X

X X

X X

X X

X

X X

X X

**代码：**

#include<cmath>

#include<algorithm>

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<stdio.h>

#include<string.h>

using namespace std;

int main(void)

{

int i,j,k,n,p,t,a[100],m=0,q;

scanf("%d",&t);

for(q=0;q<t;q++)

{

m=0;

scanf("%d",&a[q]);

n=a[q];

for(j=1;j<=a[q]/2+1;j++)

{

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

{

if(k==j||k==a[q]-j+1)

printf("X");

else

printf(" ");

}

n--;

printf("\n");

if(q==t-1)

{

m=1;

}

}

n++;

for(;j<=a[q];j++)

{

n++;

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

{

if(k==j||k==a[q]-j+1)

printf("X");

else

printf(" ");

}

printf("\n");

if(q==t-1)

{

m=1;

}

}

printf("\n");

}

}

**题目描述**

辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”

如果你是辰辰，你能完成这个任务吗？

**输入格式：**

第一行有222个整数T(1≤T≤1000)T(1 \le T \le 1000)*T*(1≤*T*≤1000)和M(1≤M≤100)M(1 \le M \le 100)*M*(1≤*M*≤100)，用一个空格隔开，TT*T*代表总共能够用来采药的时间，MM*M*代表山洞里的草药的数目。  
接下来的MM*M*行每行包括两个在111到100100100之间（包括111和100100100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

**输出格式：**

111个整数，表示在规定的时间内可以采到的草药的最大总价值。

**输入输出样例**

**输入样例#1：**

70 3

71 100

69 1

1 2

**输出样例#1：**

3

**说明**

对于30%的数据，M≤10M \le 10*M*≤10；

对于全部的数据，M≤100M \le 100*M*≤100。

**代码：**

#include<cmath>

#include<algorithm>

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<stdio.h>

#include<string.h>

using namespace std;

int main(void)

{

int i=0,j=0,t,m;

int yao[101][2]={0};

int qian[101][1001]={0};

scanf("%d%d",&t,&m);

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

{

scanf("%d%d",&yao[i][0],&yao[i][1]);

}

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

{

for(j=1;j<=t;j++)

{

if(j>=yao[i][0])

{

if(yao[i][1]+qian[i-1][j-yao[i][0]]>qian[i-1][j])

qian[i][j]=yao[i][1]+qian[i-1][j-yao[i][0]];

else

qian[i][j]=qian[i-1][j];

}

else

qian[i][j]=qian[i-1][j];

}

}

printf("%d",qian[m][t]);

return 0;

}

**题目描述**

LiYuxiang是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同种类的草药，采每一种都需要一些时间，每一种也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”

如果你是LiYuxiang，你能完成这个任务吗？

此题和原题的不同点：

1.每种草药可以无限制地疯狂采摘。

2.药的种类眼花缭乱，采药时间好长好长啊！师傅等得菊花都谢了！

**输入格式：**

输入第一行有两个整数T（1 <= T <= 100000）和M（1 <= M <= 10000），用一个空格隔开，T代表总共能够用来采药的时间，M代表山洞里的草药的数目。接下来的M行每行包括两个在1到10000之间（包括1和10000）的整数，分别表示采摘某种草药的时间和这种草药的价值。

**输出格式：**

输出一行，这一行只包含一个整数，表示在规定的时间内，可以采到的草药的最大总价值。

**输入样例#1：**

70 3

71 100

69 1

1 2

**输出样例#1：**

140

**说明**

对于30%的数据，M <= 1000；

对于全部的数据，M <= 10000，且M\*T<10000000(别数了，7个0)。

**代码：**

#include<cmath>

#include<algorithm>

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<stdio.h>

#include<string.h>

using namespace std;

int main(void)

{

int i=0,j=0,m,t;

int yao[10050][2]={0};

int x[100050]={0};

scanf("%d%d",&t,&m);

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

{

scanf("%d%d",&yao[i][0],&yao[i][1]);

}

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

{

for(j=1;j<=t;j++)

{

if(j>=yao[i][0])

{

if(yao[i][1]+x[j-yao[i][0]]>x[j])

x[j]=yao[i][1]+x[j-yao[i][0]];

}

}

}

printf("%d\n",x[t]);

return 0;

}

输入一个英文句子，将每个单词的第一个字母改成大写字母。

**Input**

输入数据包含多个测试实例，每个测试实例是一个长度不超过100的英文句子，占一行。

**Output**

请输出按照要求改写后的英文句子。

**Sample Input**

i like acm

i want to get an accepted

**Sample Output**

I Like Acm

I Want To Get An Accepted

代码：

#include<stdio.h>

int main(void)

{

char a[200];

int i;

while(gets(a)!= NULL)

{

if(a[0]>96&&a[0]<123)

a[0]=a[0]-32;

for(i=1;a[i]!='\0';i++)

{

if(a[i]==' ')

a[i+1]=a[i+1]-32;

}

for(i=0;a[i]!='\0';i++)

printf("%c",a[i]);

printf("\n");

}

}

# A + B Problem II

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 464470    Accepted Submission(s): 89927**

**Problem Description**

I have a very simple problem for you. Given two integers A and B, your job is to calculate the Sum of A + B.

**Input**

The first line of the input contains an integer T(1<=T<=20) which means the number of test cases. Then T lines follow, each line consists of two positive integers, A and B. Notice that the integers are very large, that means you should not process them by using 32-bit integer. You may assume the length of each integer will not exceed 1000.

**Output**

For each test case, you should output two lines. The first line is "Case #:", # means the number of the test case. The second line is the an equation "A + B = Sum", Sum means the result of A + B. Note there are some spaces int the equation. Output a blank line between two test cases.

**Sample Input**

2

1 2

112233445566778899 998877665544332211

**Sample Output**

Case 1:

1 + 2 = 3

Case 2:

112233445566778899 + 998877665544332211 = 1111111111111111110

代码：

*#include<cmath>*

*#include<algorithm>*

*#include<iostream>*

*#include<cstdio>*

*#include<cstdlib>*

*#include<stdio.h>*

*#include<string.h>*

*using namespace std;*

int big(int m,int n)

{

int i,l1,l2,s1,s2,j,s=0;

char b[1000],a[1000],c[1005]={0};

scanf("%s",&a);

scanf("%s",&b);

l1=strlen(a)-1;

s1=strlen(a)-1;

l2=strlen(b)-1;

s2=strlen(b)-1;

for(j=1004;l1>=0||l2>=0;)

{

if(l1>=0&&l2>=0)

{

if(57<a[l1]+b[l2]+s-'0')

{

c[j]=a[l1]+b[l2]+s-10-'0';

s=1;

}

else

{

c[j]=a[l1]+b[l2]+s-'0';

s=0;

}

j--;

l1--;

l2--;

}

if(l1>=0&&l2==-1)

{

if(57<a[l1]+s)

{

c[j]=a[l1]+s-10;

s=1;

}

else

{

c[j]=a[l1]+s;

s=0;

}

j--;

l1--;

}

if(l1==-1&&l2>=0)

{

if(57<b[l2]+s)

{

c[j]=b[l2]+s-10;

s=1;

}

else

{

c[j]=b[l2]+s;

s=0;

}

j--;

l2--;

}

}

printf("Case %d:\n",m);

for(i=0;i<=s1;i++)

{

printf("%c",a[i]);

}

printf(" + ");

for(i=0;i<=s2;i++)

{

printf("%c",b[i]);

}

printf(" = ");

for(j+=1;j<1005;j++)

{

printf("%c",c[j]);

}

printf("\n");

if(n!=0)

{

printf("\n");

}

return 0;

}

int main (void)

{

int m=0,n;

scanf("%d",&n);

while(n!=0)

{

n--;

m++;

big(m,n);

}

}

# gnatius and the Princess II

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 12600    Accepted Submission(s): 7216**

**Problem Description**

Now our hero finds the door to the BEelzebub feng5166. He opens the door and finds feng5166 is about to kill our pretty Princess. But now the BEelzebub has to beat our hero first. feng5166 says, "I have three question for you, if you can work them out, I will release the Princess, or you will be my dinner, too." Ignatius says confidently, "OK, at last, I will save the Princess."  
  
"Now I will show you the first problem." feng5166 says, "Given a sequence of number 1 to N, we define that 1,2,3...N-1,N is the smallest sequence among all the sequence which can be composed with number 1 to N(each number can be and should be use only once in this problem). So it's easy to see the second smallest sequence is 1,2,3...N,N-1. Now I will give you two numbers, N and M. You should tell me the Mth smallest sequence which is composed with number 1 to N. It's easy, isn't is? Hahahahaha......"  
Can you help Ignatius to solve this problem?

**Input**

The input contains several test cases. Each test case consists of two numbers, N and M(1<=N<=1000, 1<=M<=10000). You may assume that there is always a sequence satisfied the BEelzebub's demand. The input is terminated by the end of file.

**Output**

For each test case, you only have to output the sequence satisfied the BEelzebub's demand. When output a sequence, you should print a space between two numbers, but do not output any spaces after the last number.

**Sample Input**

6 4

11 8

**Sample Output**

1 2 3 5 6 4

1 2 3 4 5 6 7 9 8 11 10

#define \_CRT\_SBCURE\_NO\_DEPRECATE

#include <set>

#include <cmath>

#include <queue>

#include <stack>

#include <vector>

#include <string>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <functional>

using namespace std;

int main()

{

int i,n,m,t,flag;

int ans[1005];

while(scanf("%d%d",&n,&m)!=EOF)

{

flag=0;

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

{

ans[i]=i+1;

}

t=n;

n=0;

m=m-1;

do

{

if(n == m)

{

for(int i=0;i<t;++i)

{

if(flag!=0)

cout<<" ";

cout<<ans[i];

flag=1;

}

cout<<endl;

break;

}

n++;

}while(next\_permutation(ans,ans+t));

}

return 0;

}

# 杨辉三角

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 103728    Accepted Submission(s): 41680**

**Problem Description**

还记得中学时候学过的杨辉三角吗？具体的定义这里不再描述，你可以参考以下的图形：  
1  
1 1  
1 2 1  
1 3 3 1  
1 4 6 4 1  
1 5 10 10 5 1

**Input**

输入数据包含多个测试实例，每个测试实例的输入只包含一个正整数n（1<=n<=30），表示将要输出的杨辉三角的层数。

**Output**

对应于每一个输入，请输出相应层数的杨辉三角，每一层的整数之间用一个空格隔开，每一个杨辉三角后面加一个空行。

**Sample Input**

2 3

**Sample Output**

1

1 1

1

1 1

1 2 1

#include<stdio.h>

int main()

{

int n;

while (scanf("%d", &n) != EOF) {

int a[50][50];

for (int i = 0; i < n; i++)

{

a[i][0] = 1;

if (i!=0)

{

a[i][i] = 1;

}

}

printf("1\n");

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

{

printf("1");

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

{

a[i][j] = a[i - 1][j - 1] + a[i - 1][j];

printf(" %d", a[i][j]);

}

printf(" 1\n");

}

printf("\n");

}

return 0;

}

# Switch Game

**Time Limit: 1000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 25539    Accepted Submission(s): 15574**

**Problem Description**

There are many lamps in a line. All of them are off at first. A series of operations are carried out on these lamps. On the i-th operation, the lamps whose numbers are the multiple of i change the condition ( on to off and off to on ).

**Input**

Each test case contains only a number n ( 0< n<= 10^5) in a line.

**Output**

Output the condition of the n-th lamp after infinity operations ( 0 - off, 1 - on ).

**Sample Input**

1

5

**Sample Output**

1

0

***Hint***

hint

Consider the second test case:

The initial condition : 0 0 0 0 0 …

After the first operation : 1 1 1 1 1 …

After the second operation : 1 0 1 0 1 …

After the third operation : 1 0 0 0 1 …

After the fourth operation : 1 0 0 1 1 …

After the fifth operation : 1 0 0 1 0 …

#include<stdio.h>

int main()

{

int n,i,sum;

while(scanf("%d",&n)!=EOF)

{

sum=0;

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

{

if((n%i)==0)

sum++;

}

if((sum%2)==0)

printf("0\n");

else

printf("1\n");

}

return 0;

}

# 人见人爱A-B

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 114517    Accepted Submission(s): 31703**

**Problem Description**

参加过上个月月赛的同学一定还记得其中的一个最简单的题目，就是{A}+{B}，那个题目求的是两个集合的并集，今天我们这个A-B求的是两个集合的差，就是做集合的减法运算。（当然，大家都知道集合的定义，就是同一个集合中不会有两个相同的元素，这里还是提醒大家一下）  
  
呵呵，很简单吧？

**Input**

每组输入数据占1行,每行数据的开始是2个整数n(0<=n<=100)和m(0<=m<=100),分别表示集合A和集合B的元素个数，然后紧跟着n+m个元素，前面n个元素属于集合A，其余的属于集合B. 每个元素为不超出int范围的整数,元素之间有一个空格隔开.  
如果n=0并且m=0表示输入的结束，不做处理。

**Output**

针对每组数据输出一行数据,表示A-B的结果,如果结果为空集合，则输出“NULL”,否则从小到大输出结果,为了简化问题，每个元素后面跟一个空格.

**Sample Input**

3 3 1 2 3 1 4 7

3 7 2 5 8 2 3 4 5 6 7 8

0 0

**Sample Output**

2 3

NULL

#define \_CRT\_SBCURE\_NO\_DEPRECATE

#include <set>

#include <cmath>

#include <queue>

#include <stack>

#include <vector>

#include <string>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <functional>

using namespace std;

int main()

{

int n,m,i,j,s;

int a[105][2],b[105][2],c[105];

while(scanf("%d%d",&n,&m)&&(n!=0||m!=0))

{

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

{

scanf("%d",&a[i][0]);

a[i][1]=1;

}

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

{

scanf("%d",&b[i][0]);

b[i][1]=1;

}

for(i=0;i<n-1;i++)

{

s=a[i][0];

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

{

if(s==a[j][0])

a[j][1]=0;

}

}

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

{

for(j=0;j<m;j++)

{

if(a[i][0]==b[j][0])

a[i][1]=0;

}

}

j=0;

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

{

if(a[i][1]==1)

{

c[j]=a[i][0];

j++;

}

}

if(j==0)

printf("NULL\n");

else

{

sort(c,c+j);

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

printf("%d ",c[i]);

printf("\n");

}

}

return 0;

}

# {A} + {B}

**Time Limit: 10000/5000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 32843    Accepted Submission(s): 13196**

**Problem Description**

给你两个集合，要求{A} + {B}.  
注:同一个集合中不会有两个相同的元素.

**Input**

每组输入数据分为三行,第一行有两个数字n,m(0<n,m<=10000),分别表示集合A和集合B的元素个数.后两行分别表示集合A和集合B.每个元素为不超出int范围的整数,每个元素之间有一个空格隔开.

**Output**

针对每组数据输出一行数据,表示合并后的集合,要求从小到大输出,每个元素之间有一个空格隔开.

**Sample Input**

1 2

1

2 3

1 2

1

1 2

**Sample Output**

1 2 3

1 2

#define \_CRT\_SBCURE\_NO\_DEPRECATE

#include <set>

#include <cmath>

#include <queue>

#include <stack>

#include <vector>

#include <string>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <functional>

using namespace std;

int main()

{

int i,k,n,m,flag,j,l;

while(scanf("%d%d",&n,&m)!=EOF)

{

int c[20010]={};

flag=0;

l=0;

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

{

flag=0;

scanf("%d",&k);

for(j=0;j<l;j++)

{

if(c[j]==k)

flag=1;

}

if(flag==0)

{

c[l]=k;

l++;

}

}

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

{

flag=0;

scanf("%d",&k);

for(j=0;j<l;j++)

{

if(c[j]==k)

flag=1;

}

if(flag==0)

{

c[l]=k;

l++;

}

}

flag=0;

sort(c,c+l);

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

{

if(flag==0)

{

printf("%d",c[i]);

flag++;

}

else

printf(" %d",c[i]);

}

printf("\n");

}

return 0;

}

# A == B

**Time Limit: 1000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 143126    Accepted Submission(s): 23148**

**Problem Description**

Give you two numbers A and B, if A is equal to B, you should print "YES", or print "NO".

**Input**

each test case contains two numbers A and B.

**Output**

for each case, if A is equal to B, you should print "YES", or print "NO".

**Sample Input**

1 2

2 2

3 3

4 3

**Sample Output**

NO

YES

YES

NO

#define \_CRT\_SBCURE\_NO\_DEPRECATE

#include <set>

#include <cmath>

#include <queue>

#include <stack>

#include <vector>

#include <string>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <functional>

using namespace std;

char a[1000005],b[1000005];

int main()

{

int l1,l2,i,flag;

while(scanf("%s%s",a,b)!=EOF)

{

flag=0;

l1=strlen(a);

l2=strlen(b);

for(i=0;i<l1;i++)

{

if(a[i]=='.')

{

flag=1;

}

}

for(i=l1-1;i>0&&flag==1;i--)

{

if(a[i]=='0')

l1--;

else

break;

}

if(l1>1&&a[l1-1]=='.')

l1--;

flag=0;

for(i=0;i<l2;i++)

{

if(b[i]=='.')

{

flag=1;

}

}

for(i=l2-1;i>0&&flag==1;i--)

{

if(b[i]=='0')

l2--;

else

break;

}

if(l2>1&&a[l2-1]=='.')

l2--;

flag=0;

if(l1==l2)

{

for(i=0;i<l1;i++)

{

if(a[i]!=b[i])

{

flag=1;

}

}

if(flag==0)

printf("YES\n");

else

printf("NO\n");

}

else

printf("NO\n");

}

return 0;

}

# Digital Roots

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 100866    Accepted Submission(s): 31342**

**Problem Description**

The digital root of a positive integer is found by summing the digits of the integer. If the resulting value is a single digit then that digit is the digital root. If the resulting value contains two or more digits, those digits are summed and the process is repeated. This is continued as long as necessary to obtain a single digit.  
  
For example, consider the positive integer 24. Adding the 2 and the 4 yields a value of 6. Since 6 is a single digit, 6 is the digital root of 24. Now consider the positive integer 39. Adding the 3 and the 9 yields 12. Since 12 is not a single digit, the process must be repeated. Adding the 1 and the 2 yeilds 3, a single digit and also the digital root of 39.

**Input**

The input file will contain a list of positive integers, one per line. The end of the input will be indicated by an integer value of zero.

**Output**

For each integer in the input, output its digital root on a separate line of the output.

**Sample Input**

24

39

0

**Sample Output**

6

3

#define \_CRT\_SBCURE\_NO\_DEPRECATE

#include <set>

#include <cmath>

#include <queue>

#include <stack>

#include <vector>

#include <string>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <functional>

using namespace std;

int main()

{

int m,s,l1,i;

char a[10005];

while(scanf("%s",a)!=EOF&&a[0]!='0')

{

s=0;

l1=strlen(a);

for(i=0;i<l1;i++)

{

m=a[i]-'0';

s=m+s;

while(s>9)

{

s=s%10+s/10;

}

}

printf("%d\n",s);

}

return 0;

}

# 排序

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 82079    Accepted Submission(s): 24805**

**Problem Description**

输入一行数字，如果我们把这行数字中的‘5’都看成空格，那么就得到一行用空格分割的若干非负整数（可能有些整数以‘0’开头，这些头部的‘0’应该被忽略掉，除非这个整数就是由若干个‘0’组成的，这时这个整数就是0）。  
  
你的任务是：对这些分割得到的整数，依从小到大的顺序排序输出。

**Input**

输入包含多组测试用例，每组输入数据只有一行数字（数字之间没有空格），这行数字的长度不大于1000。    
  
输入数据保证：分割得到的非负整数不会大于100000000；输入数据不可能全由‘5’组成。

**Output**

对于每个测试用例，输出分割得到的整数排序的结果，相邻的两个整数之间用一个空格分开，每组输出占一行。

**Sample Input**

0051231232050775

**Sample Output**

0 77 12312320

#define \_CRT\_SBCURE\_NO\_DEPRECATE

#include <set>

#include <cmath>

#include <queue>

#include <stack>

#include <vector>

#include <string>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <functional>

using namespace std;

int main()

{

int p,q,o,l1,sum,k,j,i;

char a[2000];

while(scanf("%s",a)!=EOF)

{

int s[1005];

p=0;

sum=0;

j=0;

l1=strlen(a);

if(a[l1-1!='5'])

{

a[l1]='5';

l1++;

}

i=0;

while(a[i]=='5'&&a[i+1]=='5')

{

p++;

i++;

}

if(a[i]=='5'&&a[i+1]!='5')

{

i++;

p++;

}

for(;i<l1;i++)

{

sum=0;

if(a[i]=='5')

{

k=1;

q=i;

o=q-p;

q--;

while(o--)

{

sum=sum+(a[q]-'0')\*k;

k=k\*10;

q--;

}

while(a[i+1]=='5')

i++;

p=i+1;

s[j++]=sum;

}

}

sort(s,s+j);

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

{

if(i==0)

printf("%d",s[i]);

else

printf(" %d",s[i]);

}

printf("\n");

}

}

斐波那契数列 （1-46）

1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,46368,75025,121393,196418,317811,514229,832040,1346269,2178309,3524578,5702887,9227465,14930352,24157817,39088169,63245986,102334155,165580141,267914296,433494437,701408733,1134903170,1836311903