

## BEGIN-1. A+B问题

### 问题描述

输入A、B，输出A+B。

说明：在“问题描述”这部分，会给出试题的意思，以及所要求的目标。

### 输入格式

输入的第一行包括两个整数，由空格分隔，分别表示A、B。

说明：“输入格式”是描述在测试你的程序时，所给的输入一定满足的格式。

做题时你应该假设所给的输入是一定满足输入格式的要求的，所以你不需要对输入的格式进行检查。多余的格式检查可能会适得其反，使用你的程序错误。

在测试的时候，系统会自动将输入数据输入到你的程序中，你不能给任何提示。比如，你在输入的时候提示“请输入A、B”之类的话是不需要的，这些多余的输出会使得你的程序被判定为错误。

### 输出格式

输出一行，包括一个整数，表示A+B的值。

说明：“输出格式”是要求你的程序在输出结果的时候必须满足的格式。

在输出时，你的程序必须满足这个格式的要求，不能少任何内容，也不能多任何内容。如果你的内容和输出格式要求的不一樣，你的程序会被判断为错误，包括你输出了提示信息、中间调试信息、计时或者统计的信息等。

### 样例输入

12 45

说明：“样例输入”给出了一组满足“输入格式”要求的输入的例子。

这里给出的输入只是可能用来测试你的程序的一个输入，在测试的时候，还会有更多的输入用来测试你的程序。

### 样例输出

57

说明：“样例输出”给出了一组满足“输出格式”要求的输出的例子。

样例输出中的结果是和样例输入中的是对应的，因此，你可以使用样例的输入输出简单的检查你的程序。

要特别指出的是，能够通过样例输入输出的程序并不一定是正确的程序，在测试的时候，会用很多组数据进行测试，而不局限于样例数据。有可能一个程序通过了样例数据，但测试的时候仍只能得0分，可能因为这个程序只在一些类似样例的特例中正确，而不具有通用性，再测试更多数据时会出现错误。

比如，对于本题，如果你写一个程序不管输入是什么都输入57，则样例数据是对的，但是测试其他数据，哪怕输入是1和2，这个程序也输出57，则对于其他数据这个程序都不正确。

## 数据规模与约定

$-10000 \leq A, B \leq 10000$ 。

说明：“数据规模与约定”中给出了试题中主要参数的范围。

这个范围对于解题非常重要，不同的数据范围会导致试题需要使用不同的解法来解决。比如本题中给的A、B范围不大，可以使用整型(int)来保存，如果范围更大，超过int的范围，则要考虑其他方法来保存大数。

有一些范围在方便的时候是在“问题描述”中直接给的，所以在做题时不仅要看这个范围，还要注意问题描述。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int A, B;
5      cin >> A >> B;
6      cout << A + B;
7      return 0;
8  }
```

## BEGIN-2. 序列求和

### 问题描述

求 $1+2+3+\dots+n$ 的值。

### 输入格式

输入包括一个整数n。

### 输出格式

输出一行，包括一个整数，表示 $1+2+3+\dots+n$ 的值。

### 样例输入

4

### 样例输出

10

### 样例输入

100

说明：有一些试题会给出多组样例输入输出以帮助你更好的做题。

一般在提交之前所有这些样例都需要测试通过才行，但这不代表这几组样例数据都正确了你的程序就是完全正确的，潜在的错误可能仍然导致你的得分较低。

### 样例输出

5050

## 数据规模与约定

$1 \leq n \leq 1,000,000,000$ 。

说明：请注意这里的数据规模。

本题直接的想法是直接使用一个循环来累加，然而，当数据规模很大时，这种“暴力”的方法往往会导致超时。此时你需要想想其他方法。你可以试一试，如果使用1000000000作为你的程序的输入，你的程序是不是能在规定的上面规定的时限内运行出来。

本题另一个要值得注意的地方是答案的大小不在你的语言默认的整型(int)范围内，如果使用整型来保存结果，会导致结果错误。

如果你使用C++或C语言而且准备使用printf输出结果，则你的格式字符串应该写成%I64d以输出long long类型的整数。

```
1  #include <iostream>
2  int main() {
3      using namespace std;
4      long long n;
5      cin >> n;
6      long long result;
7      result = (1 + n) * n / 2;
8      printf("%I64d", result);
9      return 0;
10 }
```

## BEGIN-3. 圆的面积

### 问题描述

给定圆的半径r，求圆的面积。

### 输入格式

输入包含一个整数r，表示圆的半径。

### 输出格式

输出一行，包含一个实数，四舍五入保留小数点后7位，表示圆的面积。

说明：在本题中，输入是一个整数，但是输出是一个实数。

对于实数输出的问题，请一定看清楚实数输出的要求，比如本题中要求保留小数点后7位，则你的程序必须严格的输出7位小数，输出过多或者过少的小数位数都是不行的，都会被认为错误。

实数输出的问题如果没有特别说明，舍入都是按四舍五入进行。

### 样例输入

4

### 样例输出

50.2654825

数据规模与约定

$1 \leq r \leq 10000$ 。

提示

本题对精度要求较高，请注意 $\pi$ 的值应该取较精确的值。你可以使用常量来表示 $\pi$ ，比如 $PI=3.14159265358979323$ ，也可以使用数学公式来求 $\pi$ ，比如 $PI=\text{atan}(1.0)*4$ 。

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int r;
6      cin >> r;
7      double s;
8      double PI = atan(1.0) * 4;
9      s = PI * r * r;
10     printf("%.7lf",s);
11     return 0;
12 }
```

## BEGIN-4. Fibonacci数列

问题描述

Fibonacci数列的递推公式为： $F_n=F_{n-1}+F_{n-2}$ ，其中 $F_1=F_2=1$ 。

当 $n$ 比较大时， $F_n$ 也非常大，现在我们想知道， $F_n$ 除以10007的余数是多少。

输入格式

输入包含一个整数 $n$ 。

输出格式

输出一行，包含一个整数，表示 $F_n$ 除以10007的余数。

说明：在本题中，答案是要求 $F_n$ 除以10007的余数，因此我们只要能算出这个余数即可，而不需要先计算出 $F_n$ 的准确值，再将计算的结果除以10007取余数，直接计算余数往往比先算出原数再取余简单。

样例输入

10

样例输出

55

样例输入

22

样例输出

7704

数据规模与约定

$1 \leq n \leq 1,000,000$ 。

```
1  #include <iostream>
2  using namespace std;
3  const int N = 1000001;
4  int main() {
5      int F[N];
6      F[1] = F[2] = 1;
7      for (int i = 3; i <= 1000000; i++) {
8          F[i] = (F[i - 1] + F[i - 2]) % 10007;
9      }
10     int n;
11     cin >> n;
12     cout << F[n];
13     return 0;
14 }
```

## BASIC-1. 闰年判断

问题描述

给定一个年份，判断这一年是不是闰年。

当以下情况之一满足时，这一年是闰年：

1. 年份是4的倍数而不是100的倍数；
2. 年份是400的倍数。

其他的年份都不是闰年。

输入格式

输入包含一个整数y，表示当前的年份。

输出格式

输出一行，如果给定的年份是闰年，则输出yes，否则输出no。

说明：当试题指定你输出一个字符串作为结果（比如本题的yes或者no，你需要严格按照试题中给定的大小写，写错大小写将不得分。

样例输入

2013

样例输出

no

样例输入

2016

样例输出

yes

数据规模与约定

1990 ≤ y ≤ 2050。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int y;
5      cin >> y;
6      if ((y % 4 == 0 && y % 100 != 0) || y % 400 == 0)
7          cout << "yes";
8      else
9          cout << "no";
10     return 0;
11 }
```

## BASIC-2. 01字串

问题描述

对于长度为5位的一个01串，每一位都可能是0或1，一共有32种可能。它们的前几个是：

00000

00001

00010

00011

00100

请按从小到大的顺序输出这32种01串。

输入格式

本试题没有输入。

输出格式

输出32行，按从小到大的顺序每行一个长度为5的01串。

样例输出

00000

00001

00010

00011

<以下部分省略>

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      for (int i = 0; i <= 1; i++)
5          for (int j = 0; j <= 1; j++)
6              for (int k = 0; k <= 1; k++)
7                  for (int l = 0; l <= 1; l++)
8                      for (int m = 0; m <= 1; m++) {
9                          cout << i << j << k << l << m << endl;
10                     }
11     return 0;
12 }
```

### BASIC-3. 字母图形

问题描述

利用字母可以组成一些美丽的图形，下面给出了一个例子：

ABCDEFGF

BABCDEF

CBABCDE

DCBABCD

EDCBABC

这是一个5行7列的图形，请找出这个图形的规律，并输出一个n行m列的图形。

输入格式

输入一行，包含两个整数n和m，分别表示你要输出的图形的行数的列数。

输出格式

输出n行，每个m个字符，为你的图形。

样例输入

5 7

样例输出

ABCDEFGF

BABCDEF

CBABCDE

DCBABCD

EDCBABC

数据规模与约定

$1 \leq n, m \leq 26$ 。

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int n, m;
6      cin >> n >> m;
7      for (int i = 0; i < n; i++) {
8          for (int j = 0; j < m; j++) {
9              cout << char('A' + abs(i - j));
10             }
11             cout << endl;
12         }
13         return 0;
14     }
```

## BASIC-4. 数列特征

问题描述

给出n个数，找出这n个数的最大值，最小值，和。

输入格式

第一行为整数n，表示数的个数。

第二行有n个数，为给定的n个数，每个数的绝对值都小于10000。

输出格式

输出三行，每行一个整数。第一行表示这些数中的最大值，第二行表示这些数中的最小值，第三行表示这些数的和。

样例输入

```
5
1 3 -2 4 5
```

样例输出

```
5
-2
11
```

数据规模与约定

$1 \leq n \leq 10000$ 。

```
1  #include <iostream>
```



```

2   using namespace std;
3   int main() {
4       int n;
5       cin >> n;
6       int a[n];
7       for (int i = 0; i < n; i++)
8           cin >> a[i];
9       int min = a[0];
10      int max = a[0];
11      int sum = a[0];
12      for (int i = 1; i < n; i++) {
13          if (a[i] < min)
14              min = a[i];
15          if (a[i] > max)
16              max = a[i];
17          sum += a[i];
18      }
19      cout << max << endl;
20      cout << min << endl;
21      cout << sum << endl;
22      return 0;
23  }

```

## BASIC-5. 查找整数

### 问题描述

给出一个包含n个整数的数列，问整数a在数列中的第一次出现是第几个。

### 输入格式

第一行包含一个整数n。

第二行包含n个非负整数，为给定的数列，数列中的每个数都不大于10000。

第三行包含一个整数a，为待查找的数。

### 输出格式

如果a在数列中出现了，输出它第一次出现的位置(位置从1开始编号)，否则输出-1。

### 样例输入

```

6
1 9 4 8 3 9
9

```

### 样例输出

```

2

```

### 数据规模与约定

1 <= n <= 1000。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int a[n];
7      for (int i = 0; i < n; i++)
8          cin >> a[i];
9      int t;
10     cin >> t;
11     int i;
12     for (i = 0; i < n; i++)
13         if (t == a[i]) break;
14     if (i == n)
15         cout << "-1";
16     else
17         cout << i + 1;
18     return 0;
19 }
```

## BASIC-6. 杨辉三角形

### 问题描述

杨辉三角形又称Pascal三角形，它的第 $i+1$ 行是 $(a+b)^i$ 的展开式的系数。

它的一个重要性质是：三角形中的每个数字等于它两肩上的数字相加。

下面给出了杨辉三角形的前4行：

```
1
1 1
1 2 1
1 3 3 1
```

给出 $n$ ，输出它的前 $n$ 行。

### 输入格式

输入包含一个数 $n$ 。

### 输出格式

输出杨辉三角形的前 $n$ 行。每一行从这一行的第一个数开始依次输出，中间使用一个空格分隔。请不要在前面输出多余的空格。

### 样例输入

4

样例输出

1

1 1

1 2 1

1 3 3 1

数据规模与约定

$1 \leq n \leq 34$ 。

```
1  #include <iostream>
2  using namespace std;
3  const int N = 40;
4  int main() {
5      int a[N][N];
6      int n;
7      cin >> n;
8      for (int i = 0; i < n; i++) {
9          a[i][0] = a[i][i] = 1;
10         for (int j = 1; j < i; j++) {
11             a[i][j] = a[i - 1][j - 1] + a[i - 1][j];
12         }
13     }
14     for (int i = 0; i < n; i++) {
15         for (int j = 0; j <= i; j++) {
16             cout << a[i][j] << " ";
17         }
18         cout << endl;
19     }
20     return 0;
21 }
```

## BASIC-7. 特殊的数字

问题描述

153是一个非常特殊的数，它等于它的每位数字的立方和，即 $153=1*1*1+5*5*5+3*3*3$ 。编程求所有满足这种条件的三位十进制数。

输出格式

按从小到大的顺序输出满足条件的三位十进制数，每个数占一行。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      for (int i = 1; i <= 9; i++) {
5          for (int j = 0; j <= 9; j++) {
6              for (int k = 0; k <= 9; k++) {
7                  if((i * i * i + j * j * j + k * k * k) == (i * 100 + j
* 10 + k))
8                      cout << i << j << k << endl;
9              }
10         }
11     }
12     return 0;
13 }

```

## BASIC-8. 回文数

### 问题描述

1221是一个非常特殊的数，它从左边读和从右边读是一样的，编程求所有这样的四位十进制数。

### 输出格式

按从小到大的顺序输出满足条件的四位十进制数。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      for (int i = 1; i <= 9; i++) {
5          for (int j = 0; j <= 9; j++) {
6              cout << i << j << j << i << endl;
7          }
8      }
9      return 0;
10 }

```

## BASIC-9. 特殊回文数

### 问题描述

123321是一个非常特殊的数，它从左边读和从右边读是一样的。

输入一个正整数n，编程求所有这样的五位和六位十进制数，满足各位数字之和等于n。

### 输入格式

输入一行，包含一个正整数n。

### 输出格式

按从小到大的顺序输出满足条件的整数，每个整数占一行。

样例输入

52

样例输出

899998

989989

998899

数据规模和约定

$1 \leq n \leq 54$ 。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      for (int i1 = 1; i1 <= 9; i1++)
7          for (int i2 = 0; i2 <= 9; i2++)
8              for (int i3 = 0; i3 <= 9; i3++) {
9                  if ((i1 + i2 + i3 + i2 + i1) == n)
10                     cout << i1 << i2 << i3 << i2 << i1 << endl;
11             }
12     for (int i1 = 1; i1 <= 9; i1++)
13         for (int i2 = 0; i2 <= 9; i2++)
14             for (int i3 = 0; i3 <= 9; i3++) {
15                 if ((i1 + i2 + i3 + i3 + i2 + i1) == n)
16                     cout << i1 << i2 << i3 << i3 << i2 << i1 << endl;
17             }
18     return 0;
19 }
```

## BASIC-10. 十进制转十六进制

问题描述

十六进制数是在程序设计时经常要使用到的一种整数的表示方式。它有0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F共16个符号，分别表示十进制数的0至15。十六进制的计数方法是满16进1，所以十进制数16在十六进制中是10，而十进制的17在十六进制中是11，以此类推，十进制的30在十六进制中是1E。

给出一个非负整数，将它表示成十六进制的形式。

输入格式

输入包含一个非负整数a，表示要转换的数。 $0 \leq a \leq 2147483647$

输出格式

输出这个整数的16进制表示

样例输入

30

样例输出

1E

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a;
5      cin >> a;
6      int i = 0;
7      char b[100];
8      if (a == 0) {
9          cout << "0";
10     } else {
11         while (a != 0) {
12             if (a % 16 >= 10)
13                 b[i++] = a % 16 + 'A' - 10;
14             else
15                 b[i++] = a % 16 + '0';
16             a = a / 16;
17         }
18     }
19
20     for (int k = i - 1; k >= 0; k--)
21         cout << b[k];
22     return 0;
23 }
```

## BASIC-11. 十六进制转十进制

问题描述

从键盘输入一个不超过8位的正的十六进制数字字符串，将它转换为正的十进制数后输出。

注：十六进制数中的10~15分别用大写的英文字母A、B、C、D、E、F表示。

样例输入

FFFF

样例输出

65535

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
```

```

5     string s;
6     while (cin >> s) {
7         int len = s.length();
8         long long sum = 0;
9         for (int i = 0; i < len; i++) {
10            if (s[i] >= 'A' && s[i] <= 'F') {
11                sum = sum * 16 + s[i] - 'A' + 10;
12            }
13            else {
14                sum = sum * 16 + s[i] - '0';
15            }
16        }
17        cout << sum;
18    }
19    return 0;
20 }

```

## BASIC-13. 数列排序

### 问题描述

给定一个长度为 $n$ 的数列，将这个数列按从小到大的顺序排列。 $1 \leq n \leq 200$

### 输入格式

第一行为一个整数 $n$ 。

第二行包含 $n$ 个整数，为待排序的数，每个整数的绝对值小于10000。

### 输出格式

输出一行，按从小到大的顺序输出排序后的数列。

### 样例输入

5

8 3 6 4 9

### 样例输出

3 4 6 8 9

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  int cmp(int a, int b) {
6      return a < b;
7  }
8
9  int main() {
10     int n;

```

```

11     int a[200];
12     cin >> n;
13     for (int i = 0; i < n; i++) {
14         cin >> a[i];
15     }
16     sort(a, a + n, cmp);
17     cout << a[0];
18     for (int i = 1; i < n; i++)
19         cout << " " << a[i];
20     return 0;
21 }

```

## BASIC-14. 时间转换

### 问题描述

给定一个以秒为单位的时间 $t$ ，要求用“<H>:<M>:<S>”的格式来表示这个时间。<H>表示时间，<M>表示分钟，而<S>表示秒，它们都是整数且没有前导的“0”。例如，若 $t=0$ ，则应输出是“0:0:0”；若 $t=3661$ ，则输出“1:1:1”。

### 输入格式

输入只有一行，是一个整数 $t$  ( $0 \leq t \leq 86399$ )。

### 输出格式

输出只有一行，是以“<H>:<M>:<S>”的格式所表示的时间，不包括引号。

### 样例输入

0

### 样例输出

0:0:0

### 样例输入

5436

### 样例输出

1:30:36

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int h = 0, m = 0, s = 0, t;
6      cin >> t;
7      h = t / 3600;
8      t = t % 3600;
9      m = t / 60;
10     t = t % 60;

```



```

11     s = t;
12     cout << h << ":" << m << ":" << s;
13     return 0;
14 }

```

## BASIC-15. 字符串对比

### 问题描述

给定两个仅由大写字母或小写字母组成的字符串(长度介于1到10之间)，它们之间的关系是以下4中情况之一：

- 1：两个字符串长度不等。比如 Beijing 和 Hebei
- 2：两个字符串不仅长度相等，而且相应位置上的字符完全一致(区分大小写)，比如 Beijing 和 Beijing
- 3：两个字符串长度相等，相应位置上的字符仅在不区分大小写的前提下才能达到完全一致（也就是说，它并不满足情况2）。比如 beijing 和 BEIjing
- 4：两个字符串长度相等，但是即使是不区分大小写也不能使这两个字符串一致。比如 Beijing 和 Nanjing

编程判断输入的两个字符串之间的关系属于这四类中的哪一类，给出所属的类的编号。

### 输入格式

包括两行，每行都是一个字符串

### 输出格式

仅有一个数字，表明这两个字符串的关系编号

### 样例输入

BEIjing

beijing

### 样例输出

3

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      string a, b;
7      cin >> a >> b;
8      int lena = a.length();
9      int lenb = b.length();
10     if (lena != lenb) {
11         cout << 1;
12         return 0;

```

```

13     }
14     int flag = 1;
15     for (int i = 0; i < lena; i++) {
16         if (a[i] != b[i]) {
17             flag = 0;
18             break;
19         }
20     }
21     if (flag == 1) {
22         cout << 2;
23         return 0;
24     }
25     int flag2 = 1;
26     for (int i = 0; i < lena; i++) {
27         a[i] = tolower(a[i]);
28         b[i] = tolower(b[i]);
29         if (a[i] != b[i]) {
30             flag2 = 0;
31             break;
32         }
33     }
34     if (flag2 == 1) {
35         cout << 3;
36     } else {
37         cout << 4;
38     }
39     return 0;
40 }

```

## BASIC-16. 分解质因数

问题描述

求出区间[a,b]中所有整数的质因数分解。

输入格式

输入两个整数a, b。

输出格式

每行输出一个数的分解，形如 $k=a_1*a_2*a_3\dots(a_1\leq a_2\leq a_3\dots)$ ，k也是从小到大的)(具体可看样例)

样例输入

3 10

样例输出

3=3

4=2\*2

5=5

6=2\*3

7=7

8=2\*2\*2

9=3\*3

10=2\*5

提示

先筛出所有素数，然后再分解。

数据规模和约定

2<=a<=b<=10000

```
1  #include <iostream>
2  using namespace std;
3  int isprime(int n) {
4      if(n <= 1)
5          return 0;
6      else if(n == 2 || n == 3)
7          return 1;
8      else {
9          for(int i = 2; i * i < n; i++) {
10             if(n % i == 0) {
11                 return 0;
12             }
13         }
14         return 1;
15     }
16 }
17 int main() {
18     int a, b;
19     cin >> a >> b;
20     for(int i = a; i <= b; i++) {
21         int temp = i;
22         cout << i << "=";
23         int flag = 0;
24         while(temp != 1) {
25             for(int j = 2; j <= temp; j++) {
26                 if(isprime(j) && temp % j == 0) {
27                     temp = temp / j;
28                     if(flag == 1)
29                         cout << "*";
30                     cout << j;
31                     flag = 1;
32                     break;
33                 }
34             }
```

```

35     }
36     cout << endl;
37 }
38 return 0;
39 }

```

## BASIC-17. 矩阵乘法

### 问题描述

给定一个N阶矩阵A，输出A的M次幂（M是非负整数）

例如：

A =

1 2

3 4

A的2次幂

7 10

15 22

### 输入格式

第一行是一个正整数N、M（ $1 \leq N \leq 30, 0 \leq M \leq 5$ ），表示矩阵A的阶数和要求的幂数

接下来N行，每行N个绝对值不超过10的非负整数，描述矩阵A的值

### 输出格式

输出共N行，每行N个整数，表示A的M次幂所对应的矩阵。相邻的数之间用一个空格隔开

### 样例输入

2 2

1 2

3 4

### 样例输出

7 10

15 22

分析：竟然有一个！0次幂！矩阵的零次幂要输出单位矩阵！。。。 （小柳柳卒，享年20）。。。

```

1  #include <iostream>
2  using namespace std;
3  long long int b[40][40];
4  int main() {
5      int n, m;

```

```

6      cin >> n >> m;
7      long long int a[40][40];
8      long long int t[40][40];
9      for(int i = 0; i < n; i++) {
10         for(int j = 0; j < n; j++) {
11             cin >> a[i][j];
12             t[i][j] = a[i][j];
13         }
14     }
15     if(m == 0) {
16         for(int i = 0; i < n; i++) {
17             for(int j = 0; j < n; j++) {
18                 if(i != j) {
19                     cout << 0 << " ";
20                 } else {
21                     cout << 1 << " ";
22                 }
23             }
24             cout << endl;
25         }
26         return 0;
27     }
28     while(--m) {
29         for(int i = 0; i < n; i++) {
30             for(int j = 0; j < n; j++) {
31                 int k = n;
32                 while(k) {
33                     b[i][j] += t[i][k-1] * a[k-1][j];
34                     k--;
35                 }
36             }
37         }
38         for(int i = 0; i < n; i++) {
39             for(int j = 0; j < n; j++) {
40                 t[i][j] = b[i][j];
41                 b[i][j] = 0;
42             }
43         }
44     }
45     for(int i = 0; i < n; i++) {
46         for(int j = 0; j < n; j++) {
47             cout << t[i][j] << " ";
48         }
49         cout << endl;
50     }
51     return 0;
52 }

```

## BASIC-18. 矩形面积交

## 问题描述

平面上有两个矩形，它们的边平行于直角坐标系的X轴或Y轴。对于每个矩形，我们给出它的一对相对顶点的坐标，请你编程算出两个矩形的交的面积。

## 输入格式

输入仅包含两行，每行描述一个矩形。

在每行中，给出矩形的一对相对顶点的坐标，每个点的坐标都用两个绝对值不超过 $10^7$ 的实数表示。

## 输出格式

输出仅包含一个实数，为交的面积，保留到小数后两位。

## 样例输入

1 1 3 3

2 2 4 4

## 样例输出

1.00

分析：先把两个矩形的坐标都变成从左下角的到右上角的顺序~然后判断是否这两个矩形不重合~然后将所有的横坐标x排序，取中间两个的差值为重合面积的一条边的长度~然后将所有的纵坐标y排序，取中间两个的差值为重合面积的另一条边的长度~然后得重合面积的值~

```
1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  using namespace std;
5  int main() {
6      double x1, y1, x2, y2, x3, y3, x4, y4;
7      cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4;
8      if(x1 > x2)
9          swap(x1, x2);
10     if(y1 > y2)
11         swap(y1, y2);
12     if(x3 > x4)
13         swap(x3, x4);
14     if(y3 > y4)
15         swap(y3, y4);
16     if((x2 < x3 && x1 < x3) || (y2 < y3 && y1 < y3) || (x4 < x1 && x3 >
x1) || (y4 < y1 && y3 < y1)) {
17         cout << "0.00";
18         return 0;
19     }
20     double a[4] = {x1, x2, x3, x4};
21     double b[4] = {y1, y2, y3, y4};
22     sort(a, a+4);
23     sort(b, b+4);
24     double area = (a[2] - a[1]) * (b[2] - b[1]);
```

```
25     printf("%.2f", area);
26     return 0;
27 }
```

## BASIC-19. 完美的代价

### 问题描述

回文串，是一种特殊的字符串，它从左往右读和从右往左读是一样的。小龙龙认为回文串才是完美的。现在给你一个串，它不一定是回文的，请你计算最少的交换次数使得该串变成一个完美的回文串。

交换的定义是：交换两个相邻的字符

例如mamad

第一次交换 ad : mamda

第二次交换 md : madma

第三次交换 ma : madam (回文！完美！)

### 输入格式

第一行是一个整数N，表示接下来的字符串的长度( $N \leq 8000$ )

第二行是一个字符串，长度为N.只包含小写字母

### 输出格式

如果可能，输出最少的交换次数。

否则输出Impossible

### 样例输入

5

mamad

### 样例输出

3

分析：过程见代码注释部分。其中有两个注意点：

1.impossible的情况：如果有一个字符出现的次数是奇数次数，而且n是偶数，那么不可能构成回文

如果n是奇数，但是已经有一个字符出现的次数是奇数次数了，那么如果又有一个字符是奇数次数，就不可能构成回文。

2.如果n是奇数，计算中间那个字符交换的次数的時候，不需要模拟把这个数移动到中间去，因为移动到中间的话假设有一对数都在左边或者都在右边，

那么交换成回文的时候就要经过中间，就会每次把cnt多加了1，而这个1是没有必要的，因为可以所有的回文移动完了之后再把这个独立的奇数移动过去，才能保证交换次数最少。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      string s;
7      cin >> s;
8      int j = n - 1;
9      int cnt = 0; //cnt用来统计交换的次数
10     int flag = 0; //flag判断是否已经有一个单独的奇个数的字符了
11     for(int i = 0; i < j; i++) { //i指针从头遍历到倒数第二个字符
12         for(int k = j; k >= i; k--) { //k指针从后面往前一直到i寻找和s[i]相同的
s[k]
13             if(k == i) { //如果找不到相同的
14                 if(n % 2 == 0 || flag == 1) { //impossible的两种情况
15                     cout << "Impossible";
16                     return 0;
17                 }
18                 flag = 1;
19                 cnt += n / 2 - i;
20             } else if(s[k] == s[i]) {
21                 for(int l = k; l < j; l++) {
22                     swap(s[l], s[l+1]); //把s[k]换到s[j]处
23                     cnt++; //统计交换次数
24                 }
25                 j--;
26                 break;
27             }
28         }
29     }
30     cout << cnt;
31     return 0;
32 }

```

## BASIC-20. 数的读法

### 问题描述

Tom教授正在给研究生讲授一门关于基因的课程，有一件事情让他颇为头疼：一条染色体上有成千上万个碱基对，它们从0开始编号，到几百万，几千万，甚至上亿。

比如说，在对学生讲解第1234567009号位置上的碱基时，光看着数字是很难准确的念出来的。

所以，他迫切地需要一个系统，然后当他输入12 3456 7009时，会给出相应的念法：

十二亿三千四百五十六万七千零九

用汉语拼音表示为

shi er yi san qian si bai wu shi liu wan qi qian ling jiu



这样他只需要照着念就可以了。

你的任务是帮他设计这样一个系统：给定一个阿拉伯数字串，你帮他按照中文读写的规范转为汉语拼音串，相邻的两个音节用一个空格符格开。

注意必须严格按照规范，比如说“10010”读作“yi wan ling yi shi”而不是“yi wan ling shi”，“100000”读作“shi wan”而不是“yi shi wan”，“2000”读作“er qian”而不是“liang qian”。

输入格式

有一个数字串，数值大小不超过2,000,000,000。

输出格式

是一个由小写英文字母，逗号和空格组成的字符串，表示该数的英文读法。

样例输入

1234567009

样例输出

shi er yi san qian si bai wu shi liu wan qi qian ling jiu

```
1  #include <iostream>
2  using namespace std;
3  string b[10] = {"ling ", "yi ", "er ", "san ", "si ", "wu ", "liu ", "qi
   ", "ba ", "jiu "};
4  string func(string t) {
5      string ans;
6      int lent = t.length();
7      if(lent >= 4) {
8          ans += b[t[0]-'0'];
9          ans += "qian ";
10     }
11     if(lent >= 3) {
12         if(t[lent-3] != '0') {
13             ans += b[t[lent-3]-'0'];
14             ans += "bai ";
15         }
16     }
17     if(lent >= 2) {
18         if(t[lent-2] == '1') {
19             if(lent != 2) {
20                 ans += "yi shi ";
21             } else {
22                 ans += "shi ";
23             }
24         }
25         if(t[lent-2] != '0' && t[lent-2] != '1') {
26             ans += b[t[lent-2]-'0'];
27             ans += "shi ";
28         }
29     }
```

```

29     }
30     if(lent >= 1) {
31         if(t[lent-1] != '0') {
32             ans += b[t[lent-1]-'0'];
33         }
34     }
35     return ans;
36 }
37
38 int main() {
39     string s;
40     cin >> s;
41     int len = s.length();
42     string sub;
43     if(len == 10) {
44         sub = s.substr(0, 2);
45         cout << func(sub) << "yi ";
46     }
47     if(len == 9) {
48         sub = s.substr(0, 1);
49         cout << func(sub) << "yi ";
50     }
51     sub = "";
52     int flag = 0;
53     if(len >= 5) {
54         for(int i = len-8; i < len - 4; i++) {
55             if(i < 0) continue;
56             if(flag == 0 && s[i] == '0') {
57                 continue;
58             }
59             sub += s[i];
60             flag = 1;
61         }
62         if(flag == 1) {
63             cout << func(sub) << "wan ";
64         }
65         flag = 0;
66         sub = "";
67         for(int i = len-4; i < len; i++) {
68             if(flag == 0 && s[i] == '0') {
69                 continue;
70             }
71             if((i == len-1 && s[i-1] == '0') || (i == len-2 && s[i-1] ==
'0') || (i == len-3 && s[i-1] == '0'))
72                 cout << "ling ";
73             sub += s[i];
74             flag = 1;
75         }
76     }

```

```

77     if(len <= 4) {
78         sub = s;
79         flag = 1;
80     }
81     if(flag == 1) {
82         cout << func(sub);
83     }
84     return 0;
85 }

```

## BASIC-21. Sine之舞

### 问题描述

最近FJ为他的奶牛们开设了数学分析课，FJ知道若要学好这门课，必须有一个好的三角函数基本功。所以他准备和奶牛们做一个“Sine之舞”的游戏，寓教于乐，提高奶牛们的计算能力。

不妨设

$$An = \sin(1 - \sin(2 + \sin(3 - \sin(4 + \dots \sin(n)) \dots))$$

$$Sn = (\dots(A1 + n)A2 + n - 1)A3 + \dots + 2)An + 1$$

FJ想让奶牛们计算Sn的值，请你帮助FJ打印出Sn的完整表达式，以方便奶牛们做题。

### 输入格式

仅有一个数：N<201。

### 输出格式

请输出相应的表达式Sn，以一个换行符结束。输出中不得含有多余的空格或换行、回车符。

### 样例输入

3

### 样例输出

((sin(1)+3)sin(1-sin(2))+2)sin(1-sin(2+sin(3)))+1

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      string a[201];
5      string s;
6      int n;
7      cin >> n;
8      for(int i = 0; i < n; i++) {
9          if(i == 0) {
10             a[i] = "sin(1";
11             continue;
12         }
13         a[i] = a[i - 1];

```

```

14         if(i % 2 == 1) {
15             a[i] += "-";
16         } else {
17             a[i] += "+";
18         }
19         a[i] += "sin(";
20         a[i] += (char)(i + '1');
21     }
22     for(int i = 0; i < n; i++) {
23         for(int j = 0; j <= i; j++) {
24             a[i] += ")";
25         }
26     }
27     for(int i = 2; i <= n; i++) {
28         s += "(";
29     }
30     for(int i = n - 1; i >= 1; i--) {
31         s += a[n - 1 - i];
32         s += "+";
33         s += (char)(i + 1 + '0');
34         s += ")";
35     }
36     s += a[n-1];
37     s += "+1";
38     cout << s;
39     return 0;
40 }

```

## BASIC-22. FJ的字符串

### 问题描述

FJ在沙盘上写了这样一些字符串：

A1 = "A"

A2 = "ABA"

A3 = "ABACABA"

A4 = "ABACABADABACABA"

... ..

你能找出其中的规律并写所有的数列AN吗？

### 输入格式

仅有一个数：  $N \leq 26$ 。

### 输出格式

请输出相应的字符串AN，以一个换行符结束。输出中不得含有多余的空格或换行、回车符。

样例输入

3

样例输出

ABACABA

```
1  #include <iostream>
2  using namespace std;
3  string dfs(int n) {
4      if(n == 1) {
5          return "A";
6      } else {
7          return dfs(n - 1) + (char)(n + 'A' - 1) + dfs(n - 1);
8      }
9  }
10
11 int main() {
12     int n;
13     cin >> n;
14     cout << dfs(n);
15     return 0;
16 }
```

## BASIC-23. 芯片测试

问题描述

有 $n$  ( $2 \leq n \leq 20$ ) 块芯片，有好有坏，已知好芯片比坏芯片多。

每个芯片都能用来测试其他芯片。用好芯片测试其他芯片时，能正确给出被测试芯片是好还是坏。而用坏芯片测试其他芯片时，会随机给出好或是坏的测试结果（即此结果与被测试芯片实际的好坏无关）。

给出所有芯片的测试结果，问哪些芯片是好芯片。

输入格式

输入数据第一行为一个整数 $n$ ，表示芯片个数。

第二行到第 $n+1$ 行为 $n \times n$ 的一张表，每行 $n$ 个数据。表中的每个数据为0或1，在这 $n$ 行中的第 $i$ 行第 $j$ 列 ( $1 \leq i, j \leq n$ ) 的数据表示用第 $i$ 块芯片测试第 $j$ 块芯片时得到的测试结果，1表示好，0表示坏， $i=j$ 时一律为1（并不表示该芯片对本身的测试结果。芯片不能对本身进行测试）。

输出格式

按从小到大的顺序输出所有好芯片的编号

样例输入

3

1 0 1

0 1 0

1 0 1

样例输出

1 3

分析：因为超过半数的芯片是好的，所以这些芯片对于其他芯片的测试结果是正确的。

所以假设该芯片是好芯片，它除了它自身测试自身之外的其他测试结果为好的个数将超过一半。

同理，假设该芯片是坏芯片，他将有超过半数的测试结果是坏芯片。

所以只要根据每一列的所有测试结果，判断其为好芯片的总数 $\geq n/2$ 的时候，这个芯片就是好芯片。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int a[20][20];
7      for(int i = 0; i < n; i++) {
8          for(int j = 0; j < n; j++) {
9              cin >> a[i][j];
10         }
11     }
12     for(int i = 0; i < n; i++) {
13         int cnt = 0;
14         for(int j = 0; j < n; j++) {
15             if(j != i)
16                 cnt += a[j][i];
17         }
18         if(cnt >= n/2) {
19             cout << i+1 << " ";
20         }
21     }
22     return 0;
23 }
```

## BASIC-24. 龟兔赛跑预测

问题描述

话说这个世界上有各种各样的兔子和乌龟，但是研究发现，所有的兔子和乌龟都有一个共同的特点——喜欢赛跑。于是世界上各个角落都不断在发生着乌龟和兔子的比赛，小华对此很感兴趣，于是决定研究不同兔子和乌龟的赛跑。他发现，兔子虽然跑比乌龟快，但它们有众所周知的毛病——骄傲且懒惰，于是在与乌龟的比赛中，一旦任一秒结束后兔子发现自己领先 $t$ 米或以上，它们就会停下来休息 $s$ 秒。对于不同的兔子， $t$ ， $s$ 的数值是不同的，但是所有的乌龟却是一致——它们不到终点决不停止。

然而有些比赛相当漫长，全程观看会耗费大量时间，而小华发现只要在每场比赛开始后记录下兔子和乌龟的数据——兔子的速度 $v_1$ （表示每秒兔子能跑 $v_1$ 米），乌龟的速度 $v_2$ ，以及兔子对应的 $t, s$ 值，以及赛道的长度 $l$ ——就能预测出比赛的结果。但是小华很懒，不想通过手工计算推测出比赛的结果，于是他找到了你——清华大学计算机系的高才生——请求帮助，请你写一个程序，对于输入的一场比赛的数据 $v_1, v_2, t, s, l$ ，预测该场比赛的结果。

输入格式

输入只有一行，包含用空格隔开的五个正整数 $v_1, v_2, t, s, l$ ，其中 $(v_1, v_2 \leq 100; t \leq 300; s \leq 10; l \leq 10000 \text{ 且为 } v_1, v_2 \text{ 的公倍数})$

输出格式

输出包含两行，第一行输出比赛结果——一个大写字母“T”或“R”或“D”，分别表示乌龟获胜，兔子获胜，或者两者同时到达终点。

第二行输出一个正整数，表示获胜者（或者双方同时）到达终点所耗费的时间（秒数）。

样例输入

10 5 5 2 20

样例输出

D

4

样例输入

10 5 5 1 20

样例输出

R

3

样例输入

10 5 5 3 20

样例输出

T

4

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int vr, vt, t, s, l;
5     cin >> vr >> vt >> t >> s >> l;
6     int ttime = l / vt;
7     int cnttime = 0;
8     int lent = 0;
9     int lenr = 0;
```

```

10     int breaktime = 0; //统计兔子休息的时间
11     int flag = 0; //flag==0表示兔子不在休息, flag==1表示兔子正在休息
12     while(lent < 1 && lenr < 1) {
13         if(flag == 0 && lenr - lent < t || flag == 1 && breaktime >= s)
14         {
15             lenr = lenr + vr;
16             flag = 0;
17             breaktime = 0;
18         } else {
19             flag = 1;
20             breaktime++;
21         }
22         lent = lent + vt;
23         cnttime++;
24     }
25     if(lent >= 1 && lenr < 1) {
26         cout << "T" << endl;
27     } else if(lent < 1 && lenr >= 1) {
28         cout << "R" << endl;
29     } else {
30         cout << "D" << endl;
31     }
32     cout << cnttime;
33     return 0;
34 }

```

## BASIC-25. 回形取数

### 问题描述

回形取数就是沿矩阵的边取数，若当前方向上无数可取或已经取过，则左转90度。一开始位于矩阵左上角，方向向下。

### 输入格式

输入第一行是两个不超过200的正整数m, n，表示矩阵的行和列。接下来m行每行n个整数，表示这个矩阵。

### 输出格式

输出只有一行，共mn个数，为输入矩阵回形取数得到的结果。数之间用一个空格分隔，行末不要有多余的空格。

### 样例输入

```

3 3
1 2 3
4 5 6
7 8 9

```



样例输出

1 4 7 8 9 6 3 2 5

样例输入

3 2

1 2

3 4

5 6

样例输出

1 3 5 6 4 2

```
1  #include <iostream>
2  #include <memory.h>
3  using namespace std;
4  int main() {
5      int m, n;
6      cin >> m >> n;
7      int a[201][201];
8      memset(a, -1, sizeof(a));
9      int i = 0, j = 0;
10     for(i = 0; i < m; i++)
11         for(j = 0; j < n; j++)
12             cin >> a[i][j];
13     i = 0, j = 0;
14     int total = 0;
15     while(total < m * n) {
16         while(i <= m-1 && a[i][j] != -1) { //down
17             cout << a[i][j] << " ";
18             a[i][j] = -1;
19             i++;
20             total++;
21         }
22         i--;
23         j++;
24         while(j <= n-1 && a[i][j] != -1) { //right
25             cout << a[i][j] << " ";
26             a[i][j] = -1;
27             j++;
28             total++;
29         }
30         j--;
31         i--;
32         while(i >= 0 && a[i][j] != -1) { //up
33             cout << a[i][j] << " ";
34             a[i][j] = -1;
```

```

35         i--;
36         total++;
37     }
38     i++;
39     j--;
40     while(j >= 0 && a[i][j] != -1) { //left
41         cout << a[i][j] << " ";
42         a[i][j] = -1;
43         j--;
44         total++;
45     }
46     j++;
47     i++;
48 }
49 return 0;
50 }

```

## BASIC-26. 报时助手

### 问题描述

给定当前的时间，请用英文的读法将它读出来。

时间用时h和分m表示，在英文的读法中，读一个时间的方法是：

如果m为0，则将时读出来，然后加上“o'clock”，如3:00读作“three o'clock”。

如果m不为0，则将时读出来，然后将分读出来，如5:30读作“five thirty”。

时和分的读法使用的是英文数字的读法，其中0~20读作：

0:zero, 1: one, 2:two, 3:three, 4:four, 5:five, 6:six, 7:seven, 8:eight, 9:nine, 10:ten, 11:eleven, 12:twelve, 13:thirteen, 14:fourteen, 15:fifteen, 16:sixteen, 17:seventeen, 18:eighteen, 19:nineteen, 20:twenty。

30读作thirty, 40读作forty, 50读作fifty。

对于大于20小于60的数字，首先读整十的数，然后再加上个位数。如31首先读30再加1的读法，读作“thirty one”。

按上面的规则21:54读作“twenty one fifty four”，9:07读作“nine seven”，0:15读作“zero fifteen”。

### 输入格式

输入包含两个非负整数h和m，表示时间的时和分。非零的数字前没有前导0。h小于24，m小于60。

### 输出格式

输出时间时刻的英文。

### 样例输入

0 15

样例输出

zero fifteen

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int h, m;
6      cin >> h >> m;
7      string arr[24] = {"zero", "one", "two", "three", "four", "five",
8      "six", "seven", "eight", "nine", "ten", "eleven", "twelve", "thirteen",
9      "fourteen", "fifteen", "sixteen", "seventeen",
10     "eighteen", "nineteen", "twenty", "twenty one", "twenty two", "twenty
11     three"};
12     cout << arr[h] << " ";
13     if (m == 0)
14         cout << "o'clock";
15     int t = m % 10;
16     m = m / 10;
17     switch(m) {
18         case 2: cout << "twenty "; break;
19         case 3: cout << "thirty "; break;
20         case 4: cout << "forty "; break;
21         case 5: cout << "fifty "; break;
22         default: break;
23     }
24     if (m == 0 && t != 0) {
25         cout << arr[t];
26     }
27     if (m == 1) {
28         cout << arr[t + 10];
29     }
30     if (m != 0 && m != 1 && t != 0) {
31         cout << arr[t];
32     }
33     return 0;
34 }
```

## BASIC-27. 2n皇后问题

问题描述

给定一个 $n \times n$ 的棋盘，棋盘有一些位置不能放皇后。现在要向棋盘放入 $n$ 个黑皇后和 $n$ 个白皇后，使任意的两个黑皇后都不在同一行、同一列或同一条对角线上，任意的两个白皇后都不在同一行、同一列或同一条对角线上。问总共有多少种放法？ $n$ 小于等于8。

输入格式

输入的第一行为一个整数 $n$ ，表示棋盘的大小。

接下来n行，每行n个0或1的整数，如果一个整数为1，表示对应的位置可以放皇后，如果一个整数为0，表示对应的位置不可以放皇后。

输出格式

输出一个整数，表示总共有多少种放法。

样例输入

4

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

样例输出

2

样例输入

4

1 0 1 1

1 1 1 1

1 1 1 1

1 1 1 1

样例输出

0

分析：和n皇后问题一样，只不过2n皇后要用两个深度优先搜索。

调用放置白皇后的递归dfs，先放置白皇后，当每一个白皇后放置成功之后，在递归的return语句之前，

新建一个棋盘，复制原来的棋盘后并把放置了白皇后的位置置为0，调用摆放黑皇后的深度优先搜索。

当黑皇后也找到相应的解法后，cnt++;最后输出cnt的值。

```
1  #include <iostream>
2  #include <cmath>
3  #include <vector>
4  using namespace std;
5  int cnt = 0;
6
7  bool issafe(vector<vector<int> > pic, vector<int> pos, int row) {
8      for(int i = 0; i < row; i++) {
9          if(pos[i] == pos[row] || abs(i - row) == abs(pos[i] -
pos[row]))
```

```

10         return false;
11     }
12     return true;
13 }
14
15 void blackdfs(vector<vector<int> > blackpic, vector<int> blackpos, int
n, int blackrow) {
16     if(blackrow == n) {
17         cnt++;
18         return ;
19     }
20     for(blackpos[blackrow] = 0; blackpos[blackrow] < n;
blackpos[blackrow]++) {
21         if(blackpic[blackrow][blackpos[blackrow]] == 1 &&
issafe(blackpic, blackpos, blackrow)) {
22             blackdfs(blackpic, blackpos, n, blackrow + 1);
23         }
24     }
25 }
26
27
28 void dfs(vector<vector<int> > pic, vector<int> pos, int n, int row) {
29     if(row == n) {
30         vector<vector<int> > blackpic(n, vector<int>(n));
31         for(int i = 0; i < n; i++) {
32             for(int j = 0; j < n; j++) {
33                 blackpic[i][j] = pic[i][j];
34             }
35         }
36         for(int i = 0; i < n; i++) {
37             blackpic[i][pos[i]] = 0;
38         }
39         vector<int> blackpos(n);
40         blackdfs(blackpic, blackpos, n, 0);
41         return ;
42     }
43     for(pos[row] = 0; pos[row] < n; pos[row]++) {
44         if(pic[row][pos[row]] == 1 && issafe(pic, pos, row)) {
45             dfs(pic, pos, n, row + 1);
46         }
47     }
48 }
49
50 int main() {
51     int n;
52     cin >> n;
53     vector<vector<int> > pic(n, vector<int>(n));
54     vector<int> pos(n);
55     for(int i = 0; i < n; i++) {

```

```

56         for(int j = 0; j < n; j++) {
57             cin >> pic[i][j];
58         }
59     }
60     dfs(pic, pos, n, 0);
61     cout << cnt;
62     return 0;
63 }

```

## BASIC-29. 高精度加法

### 问题描述

输入两个整数a和b，输出这两个整数的和。a和b都不超过100位。

### 算法描述

由于a和b都比较大，所以不能直接使用语言中的标准数据类型来存储。对于这种问题，一般使用数组来处理。

定义一个数组A，A[0]用于存储a的个位，A[1]用于存储a的十位，依此类推。同样可以用一个数组B来存储b。

计算 $c = a + b$ 的时候，首先将A[0]与B[0]相加，如果有进位产生，则把进位（即和的十位数）存入r，把和的个位数存入C[0]，即C[0]等于(A[0]+B[0])%10。然后计算A[1]与B[1]相加，这时还应将低位进上来的值r也加起来，即C[1]应该是A[1]、B[1]和r三个数的和。如果又有进位产生，则仍可将新的进位存入到r中，和的个位存到C[1]中。依此类推，即可求出C的所有位。

最后将C输出即可。

### 输入格式

输入包括两行，第一行为一个非负整数a，第二行为一个非负整数b。两个整数都不超过100位，两数的最高位都不是0。

### 输出格式

输出一行，表示a + b的值。

### 样例输入

20100122201001221234567890

2010012220100122

### 样例输出

20100122203011233454668012

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      string a;

```

```

7     string b;
8     int A[100] = {0};
9     int B[100] = {0};
10    cin >> a >> b;
11    int lena = a.length(), lenb = b.length();
12    int j = 0;
13    for (int i = lena - 1; i >= 0; i--) {
14        A[j++] = a[i] - '0';
15    }
16    j = 0;
17    for (int i = lenb - 1; i >= 0; i--) {
18        B[j++] = b[i] - '0';
19    }
20    int C[101] = {0};
21    int temp = 0;
22    for (int i = 0; i < 100; i++) {
23        C[i] = A[i] + B[i] + temp;
24        temp = C[i] / 10;
25        C[i] = C[i] % 10;
26    }
27    int max = lena;
28    if (max < lenb)
29        max = lenb;
30    max = max - 1;
31    if (C[max + 1] != 0)
32        max = max + 1;
33    for (int i = max; i >= 0; i--) {
34        cout << C[i];
35    }
36    return 0;
37 }

```

## BASIC-30. 阶乘计算

### 问题描述

输入一个正整数 $n$ ，输出 $n!$ 的值。

其中 $n!=1*2*3*\dots*n$ 。

### 算法描述

$n!$ 可能很大，而计算机能表示的整数范围有限，需要使用高精度计算的方法。使用一个数组 $A$ 来表示一个大整数 $a$ ， $A[0]$ 表示 $a$ 的个位， $A[1]$ 表示 $a$ 的十位，依次类推。

将 $a$ 乘以一个整数 $k$ 变为将数组 $A$ 的每一个元素都乘以 $k$ ，请注意处理相应的进位。

首先将 $a$ 设为1，然后乘2，乘3，当乘到 $n$ 时，即得到了 $n!$ 的值。

### 输入格式

输入包含一个正整数 $n$ ， $n \leq 1000$ 。

输出格式

输出n!的准确值。

样例输入

10

样例输出

3628800

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n;
6      cin >> n;
7      int A[10000] = {1};
8      for (int i = 1; i <= n; i++) {
9          for (int j = 0; j < 10000; j++) {
10             A[j] = A[j] * i;
11         }
12         for (int j = 0; j < 10000; j++) {
13             if (A[j] >= 9) {
14                 A[j + 1] = A[j + 1] + A[j] / 10;
15                 A[j] = A[j] % 10;
16             }
17         }
18     }
19     int t = 0;
20     for (int i = 9999; i >= 0; i--) {
21         if (A[i] != 0) {
22             t = i;
23             break;
24         }
25     }
26     for (int i = t; i >= 0; i--) {
27         cout << A[i];
28     }
29     return 0;
30 }
```

## ALGO-1. 区间k大数查询

问题描述

给定一个序列，每次询问序列中第l个数到第r个数中第K大的数是哪个。

输入格式

第一行包含一个数n，表示序列长度。



第二行包含n个正整数，表示给定的序列。

第三个包含一个正整数m，表示询问个数。

接下来m行，每行三个数l,r,K，表示询问序列从左往右第l个数到第r个数中，从大往小第K大的数是哪个。序列元素从1开始标号。

输出格式

总共输出m行，每行一个数，表示询问的答案。

样例输入

```
5
1 2 3 4 5
2
1 5 2
2 3 2
```

样例输出

```
4
2
```

数据规模与约定

对于30%的数据， $n, m \leq 100$ ；

对于100%的数据， $n, m \leq 1000$ ；

保证 $k \leq (r - l + 1)$ ，序列中的数 $\leq 106$ 。

```
1  #include <iostream>
2  #include <algorithm>
3  #include <vector>
4  using namespace std;
5  int cmp(int a, int b){return a > b;}
6  int main() {
7      int n, m;
8      cin >> n;
9      vector<int> a(n);
10     for (int i = 0; i < n; i++) {
11         cin >> a[i];
12     }
13     cin >> m;
14     vector<int> result(m);
15     for (int i = 0; i < m; i++) {
16         int l, r, k;
17         cin >> l >> r >> k;
18         int *temp = new int [n];
19         for(int j = 0; j < n; j++) {
```

```

20         temp[j] = a[j];
21     }
22     sort(temp + l - 1, temp + r, cmp);
23     result[i] = temp[l - 1 + k - 1];
24     delete [] temp;
25 }
26 for (int i = 0; i < m; i++) {
27     cout << result[i] << endl;
28 }
29 return 0;
30 }

```

## ALGO-2. 最大最小公倍数（贪心算法）

### 问题描述

已知一个正整数N，问从1~N中任选出三个数，他们的最小公倍数最大可以为多少。

### 输入格式

输入一个正整数N。

### 输出格式

输出一个整数，表示你找到的最小公倍数。

### 样例输入

9

### 样例输出

504

### 数据规模与约定

$1 \leq N \leq 10^6$ 。

### 分析:

1.如果  $n \leq 2$ , 那么最小公倍数为  $n$

2.如果  $n$  是奇数, 那么最小公倍数的最大值为末尾的三个数相乘

3.如果是偶数的话, 如果同时出现两个偶数肯定会不能构成最大值了, 因为会被除以2~~分两种情况:

(1)如果  $n$  是偶数且不是三的倍数, 比如8, 那么跳过 $n-2$ 这个数而选择  $8\ 7\ 5$  能保证不会最小公倍数被除以2~~所以最小公倍数的最大值为  $n * (n - 1) * (n - 3)$

(2)如果  $n$  是偶数且为三的倍数, 比如6, 如果还像上面那样选择的话, 6和3相差3会被约去一个3, 又不能构成最大值了。那么最小公倍数的最大值为  $(n - 1) * (n - 2) * (n - 3)$

```

1  #include <iostream>
2  using namespace std;
3  int main() {

```

```

4     long long n, ans;
5     cin >> n;
6     if(n <= 2)
7         ans = n;
8     else if(n % 2 == 1)
9         ans = n * (n-1) * (n-2);
10    else if(n % 3 == 0)
11        ans = (n - 1) * (n - 2) * (n - 3);
12    else
13        ans = n * (n - 1) * (n - 3);
14    cout << ans;
15    return 0;
16 }

```

## ALGO-4. 结点选择

有一棵  $n$  个节点的树，树上每个节点都有一个正整数权值。如果一个点被选择了，那么在树上和它相邻的点都不能被选择。求选出的点的权值和最大是多少？

第一行包含一个整数  $n$ 。

接下来一行包含  $n$  个正整数，第  $i$  个正整数代表点  $i$  的权值。

接下来一共  $n-1$  行，每行描述树上的一条边。

对于20%的数据， $n \leq 20$ 。

对于50%的数据， $n \leq 1000$ 。

对于100%的数据， $n \leq 100000$ 。

权值均为不超过1000的正整数。

分析：题目给出的数据不一定是二叉树，所以可以看作图来处理～其实就是用邻接表存储啦～ $v[i]$ 数组中保存 $i$ 点的孩子节点们～ $dp[i][0]$ 表示不取 $i$ 点的结果～ $dp[i][1]$ 表示取 $i$ 点的结果～

用深度优先搜索+动态规划，每个点的最大权值有取当前这个点和不取当前这个点两种情况～如果取当前点，则不能取与它相邻的任何点；不取当前点，则取与它相邻点的最大值进行累加～从底向上累加到顶部～ $\max(dp[1][0], dp[1][1])$ 就是所求结果～

用一个变量 $pre$ 保存当前结点的前一个结点～如果等于 $pre$ 说明访问到了它的父亲结点，为了防止重复访问，要在 $v[node][i]$ 不等于 $pre$ 时候继续dfs下去～否则可能会形成无限循环的环～

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int dp[100010][2];
5  vector<vector<int>> > v;
6  void dfs(int node, int pre) {
7      for (int i = 0; i < v[node].size(); i++) {
8          int temp = v[node][i];
9          if (temp != pre) {
10             dfs(temp, node);

```

```

11         dp[node][0] += max(dp[temp][0], dp[temp][1]);
12         dp[node][1] += dp[temp][0];
13     }
14 }
15 }
16 int main() {
17     int n, a, b;
18     scanf("%d", &n);
19     for (int i = 1; i <= n; i++)
20         scanf("%d", &dp[i][1]);
21     v.resize(n + 1);
22     for (int i = 1; i <= n - 1; i++) {
23         scanf("%d%d", &a, &b);
24         v[a].push_back(b);
25         v[b].push_back(a);
26     }
27     dfs(1, 0);
28     cout << max(dp[1][0], dp[1][1]);
29     return 0;
30 }

```

## ALGO-5. 最短路

### 问题描述

给定一个 $n$ 个顶点， $m$ 条边的有向图（其中某些边权可能为负，但保证没有负环）。请你计算从1号点到其他点的最短路（顶点从1到 $n$ 编号）。

### 输入格式

第一行两个整数 $n, m$ 。

接下来的 $m$ 行，每行有三个整数 $u, v, l$ ，表示 $u$ 到 $v$ 有一条长度为 $l$ 的边。

### 输出格式

共 $n-1$ 行，第 $i$ 行表示1号点到 $i+1$ 号点的最短路。

### 样例输入

```

3 3
1 2 -1
2 3 -1
3 1 2

```

### 样例输出

```

-1
-2

```

### 数据规模与约定

对于10%的数据,  $n = 2$ ,  $m = 2$ 。

对于30%的数据,  $n \leq 5$ ,  $m \leq 10$ 。

对于100%的数据,  $1 \leq n \leq 20000$ ,  $1 \leq m \leq 200000$ ,  $-10000 \leq l \leq 10000$ , 保证从任意顶点都能到达其他所有顶点。

分析: 1.  $n$ 个点,  $m$ 条有向边, 含有负边

1. 外层循环 $n-1$ 次, 内层循环 $m$ 次, 进行松弛

3. 添加check变量判断本轮是否进行松弛了, 如果未进行松弛则可以提前退出循环 (否则会超时)

4. 处理有向边时, 注意 $u[i]$ 和 $v[i]$ 的顺序不要颠倒~

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int dis[20001] = {0}, u[200001], v[200001], w[200001];
5      int n, m, inf = 99999999;
6      fill(dis+2, dis+20001, inf);
7      scanf("%d %d", &n, &m);
8      for(int i = 1; i <= m; i++)
9          scanf("%d %d %d", &u[i], &v[i], &w[i]);
10     for(int k = 1; k <= n - 1; k++){
11         int check = 0;
12         for(int i = 1; i <= m; i++){
13             if(dis[v[i]] > dis[u[i]] + w[i]){
14                 dis[v[i]] = dis[u[i]] + w[i];
15                 check = 1;
16             }
17         }
18         if(check == 0) break;
19     }
20     for(int i = 2; i <= n; i++)
21         printf("%d\n", dis[i]);
22     return 0;
23 }
```

## ALGO-8. 操作格子 (线段树)

问题描述

有 $n$ 个格子, 从左到右放成一排, 编号为 $1-n$ 。

共有 $m$ 次操作, 有3种操作类型:

1. 修改一个格子的权值,
2. 求连续一段格子权值和,
3. 求连续一段格子的最大值。

对于每个2、3操作输出你所求出的结果。

输入格式

第一行2个整数n, m。

接下来一行n个整数表示n个格子的初始权值。

接下来m行, 每行3个整数p,x,y, p表示操作类型, p=1时表示修改格子x的权值为y, p=2时表示求区间[x,y]内格子权值和, p=3时表示求区间[x,y]内格子最大的权值。

输出格式

有若干行, 行数等于p=2或3的操作总数。

每行1个整数, 对应了每个p=2或3操作的结果。

样例输入

```
4 3
1 2 3 4
2 1 3
1 4 3
3 1 4
```

样例输出

```
6
3
```

数据规模与约定

对于20%的数据  $n \leq 100$ ,  $m \leq 200$ 。

对于50%的数据  $n \leq 5000$ ,  $m \leq 5000$ 。

对于100%的数据  $1 \leq n \leq 100000$ ,  $m \leq 100000$ ,  $0 \leq \text{格子权值} \leq 10000$ 。

分析: 用结构体数组建立一棵线段树~当p==1时从上到下更新这个线段树的值, 当p==2的时候搜索对应区间内的总和~当p==3的时候搜索对应区间的最大值~

```
1  #include <iostream>
2  #define max(a, b) a > b ? a : b;
3  using namespace std;
4
5  struct node {
6      int l;
7      int r;
8      int maxvalue;
9      int sum;
10 } a[1000000];
11
12 void init(int left, int right, int i) {
13     a[i].l = left;
14     a[i].r = right;
```

```

15     a[i].maxvalue = 0;
16     a[i].sum = 0;
17     if(left != right) {
18         int mid = (left + right) / 2;
19         init(left, mid, 2 * i);
20         init(mid + 1, right, 2*i+1);
21     }
22 }
23
24 void insert(int i, int j, int value) {
25     if(a[i].l == a[i].r) {
26         a[i].maxvalue = value;
27         a[i].sum = value;
28         return ;
29     }
30     int mid = (a[i].l + a[i].r) / 2;
31     if(j <= mid)
32         insert(2 * i, j, value);
33     else
34         insert(2 * i + 1, j, value);
35     a[i].maxvalue = max(a[2*i].maxvalue, a[2*i+1].maxvalue);
36     a[i].sum = a[2*i].sum + a[2*i+1].sum;
37 }
38
39 int find_sum(int i, int x, int y) {
40     if(x == a[i].l && y == a[i].r) {
41         return a[i].sum;
42     }
43     int mid = (a[i].l + a[i].r) / 2;
44     if(y <= mid)
45         return find_sum(2*i, x, y);
46     else if(x > mid)
47         return find_sum(2*i+1, x, y);
48     else
49         return find_sum(2*i, x, mid)+ find_sum(2*i+1, mid+1, y);
50 }
51
52 int find_max(int i, int x, int y) {
53     if(x == a[i].l && y == a[i].r) {
54         return a[i].maxvalue;
55     }
56     int mid = (a[i].l + a[i].r) / 2;
57     if(y <= mid)
58         return find_max(2*i, x, y);
59     else if(x > mid)
60         return find_max(2*i+1, x, y);
61     else
62         return max(find_max(2*i, x, mid), find_max(2*i+1, mid+1, y));
63 }

```

```

64
65
66 int main() {
67     int n, m;
68     cin >> n >> m;
69     init(1, n, 1);
70     int value;
71     for(int j = 1; j <= n; j++) {
72         cin >> value;
73         insert(1, j, value);
74     }
75     for(int k = 0; k < m; k++) {
76         int p, x, y;
77         cin >> p >> x >> y;
78         if(p == 1)
79             insert(1, x, y);
80         if(p == 2)
81             cout << find_sum(1, x, y) << endl;
82         if(p == 3)
83             cout << find_max(1, x, y) << endl;
84     }
85     return 0;
86 }

```

## ALGO-10. 集合运算

### 问题描述

给出两个整数集合A、B，求出他们的交集、并集以及B在A中的余集。

### 输入格式

第一行为一个整数n，表示集合A中的元素个数。

第二行有n个互不相同的用空格隔开的整数，表示集合A中的元素。

第三行为一个整数m，表示集合B中的元素个数。

第四行有m个互不相同的用空格隔开的整数，表示集合B中的元素。

集合中的所有元素均为int范围内的整数，n、m<=1000。

### 输出格式

第一行按从小到大的顺序输出A、B交集中的所有元素。

第二行按从小到大的顺序输出A、B并集中的所有元素。

第三行按从小到大的顺序输出B在A中的余集中的所有元素。

### 样例输入

```

5
1 2 3 4 5
5
2 4 6 8 10

```

### 样例输出



2 4  
1 2 3 4 5 6 8 10  
1 3 5

样例输入

4  
1 2 3 4  
3  
5 6 7

样例输出

1 2 3 4 5 6 7  
1 2 3 4

分析：1.利用map，第一个集合的元素标记为1，第二个集合的元素累加2  
2.判断，值为1的为余集，有值的为交集，值为3的为并集，且map自带排序功能~

```
1  #include <iostream>
2  #include <map>
3  using namespace std;
4  int main() {
5      int n, m, t;
6      map<int, int> a;
7      scanf("%d", &n);
8      for (int i = 0; i < n; i++) {
9          scanf("%d", &t);
10         a[t] = 1;
11     }
12     scanf("%d", &m);
13     for (int i = 0; i < m; i++) {
14         scanf("%d", &t);
15         a[t] += 2;
16     }
17     for (map<int, int>::iterator i = a.begin(); i != a.end(); i++)
18         if (i->second == 3) cout << i->first << ' ';
19     cout << endl;
20     for (map<int, int>::iterator i = a.begin(); i != a.end(); i++)
21         cout << i->first << ' ';
22     cout << endl;
23     for (map<int, int>::iterator i = a.begin(); i != a.end(); i++)
24         if (i->second == 1) cout << i->first << ' ';
25     cout << endl;
26     return 0;
27 }
```

## ALGO-11. 瓷砖铺放（递归/动态规划）

问题描述

有一长度为 $N(1 \leq N \leq 10)$ 的地板，给定两种不同瓷砖：一种长度为1，另一种长度为2，数目不限。要将这个长度为 $N$ 的地板铺满，一共有多少种不同的铺法？

例如，长度为4的地面一共有如下5种铺法：

$4=1+1+1+1$

$4=2+1+1$

$4=1+2+1$

$4=1+1+2$

$4=2+2$

编程用递归的方法求解上述问题。

输入格式

只有一个数 $N$ ，代表地板的长度

输出格式

输出一个数，代表所有不同的瓷砖铺放方法的总数

样例输入

4

样例输出

5

分析：用动态规划的方法解：

分析：用动态规划的方法解：

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int main() {
5      int n;
6      cin >> n;
7      vector<int> v(n+1);
8      v[0] = 1, v[1] = 1;
9      for(int i = 2; i <= n; i++) {
10         v[i] = v[i-1] + v[i-2];
11     }
12     cout << v[n];
13     return 0;
14 }
```

## ALGO-12. 幂方分解

## 问题描述

任何一个正整数都可以用2的幂次方表示。例如：

$$137=2^7+2^3+2^0$$

同时约定方次用括号来表示，即 $a^b$ 可表示为 $a(b)$ 。

由此可知，137可表示为：

$$2(7)+2(3)+2(0)$$

进一步：7=2<sup>2</sup>+2+2<sup>0</sup>（2<sup>1</sup>用2表示）

$$3=2+2^0$$

所以最后137可表示为：

$$2(2(2)+2+2(0))+2(2+2(0))+2(0)$$

又如：

$$1315=2^{10}+2^8+2^5+2+1$$

所以1315最后可表示为：

$$2(2(2+2(0))+2)+2(2(2+2(0)))+2(2(2)+2(0))+2+2(0)$$

## 输入格式

输入包含一个正整数N（ $N \leq 20000$ ），为要求分解的整数。

## 输出格式

程序输出包含一行字符串，为符合约定的n的0，2表示（在表示中不能有空格）

分析：递归即可，注意递归边界是1,返回"1"，最后输出的时候过滤掉即~

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  string dfs(int n) {
5      if (n == 1) return "1";
6      if (n == 0) return "0";
7      string ans;
8      vector<int> v;
9      int cnt = 0;
10     while (n) {
11         if (n % 2 == 1) v.push_back(cnt);
12         n /= 2;
13         cnt++;
14     }
15     for (int i = v.size() - 1; i >= 0; i--) {
16         ans += "2(" + dfs(v[i]) + ")";
17         if (i != 0) ans += "+";
18     }
19     return ans;
20 }
21 int main() {
22     int n;
23     cin >> n;
24     string ans = dfs(n);
25     for (int i = 0; i < ans.size(); i++) {
```

```

26         if (ans[i + 1] == '1') i += 3;
27         cout << ans[i];
28     }
29     return 0;
30 }

```

## ALGO-14. 回文数

### 问题描述

若一个数（首位不为零）从左向右读与从右向左读都一样，我们就将其称之为回文数。

例如：给定一个10进制数56，将56加65（即把56从右向左读），得到121是一个回文数。

又如：对于10进制数87：

STEP1:  $87+78 = 165$

STEP2:  $165+561 = 726$

STEP3:  $726+627 = 1353$

STEP4:  $1353+3531 = 4884$

在这里的一步是指进行了一次N进制的加法，上例最少用了4步得到回文数4884。

写一个程序，给定一个N（ $2 \leq N \leq 10$ 或 $N=16$ ）进制数M（其中16进制数字为0-9与A-F），求最少经过几步可以得到回文数。

如果在30步以内（包含30步）不可能得到回文数，则输出“Impossible!”

### 输入格式

两行，N与M

### 输出格式

如果能在30步以内得到回文数，输出“STEP=xx”（不含引号），其中xx是步数；否则输出一行“Impossible!”（不含引号）

### 样例输入

9

87

### 样例输出

STEP=6

分析：正常模拟加法即可，注意因为涉及进制超过10，过程不用string而用vector模拟~

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  int k, cnt = 0;
5  using namespace std;
6  vector<int> f(vector<int> v) {
7      int carry = 0;
8      vector<int> v1(v), v2(v), ans(v.size(), 0);
9      reverse(v1.begin(), v1.end());
10     for (int i = v.size() - 1; i >= 0; i--) {

```

```

11         ans[i] = (v1[i] + v2[i] + carry) % k;
12         carry = (v1[i] + v2[i] + carry) / k;
13     }
14     if (carry > 0) ans.insert(ans.begin(), carry);
15     return ans;
16 }
17 bool is(vector<int> v) {
18     for (int i = 0; i < v.size(); i++)
19         if (v[i] != v[v.size() - i - 1]) return false;
20     return true;
21 }
22 int main() {
23     vector<int> v;
24     string s;
25     cin >> k >> s;
26     for (int i = 0; i < s.length(); i++) {
27         if (s[i] >= '0' && s[i] <= '9') v.push_back(s[i] - '0');
28         else v.push_back(s[i] - 'A' + 10);
29     }
30     while (!is(v)) {
31         v = f(v);
32         cnt++;
33         if(cnt == 30){
34             printf("Impossible!\n");
35             return 0;
36         }
37     }
38     printf("STEP=%d\n", cnt);
39     return 0;
40 }

```

## ALGO-20. 求先序排列

### 问题描述

给出一棵二叉树的中序与后序排列。求出它的先序排列。（约定树结点用不同的大写字母表示，长度  $\leq 8$ ）。

### 输入格式

两行，每行一个字符串，分别表示中序和后序排列

### 输出格式

一个字符串，表示所求先序排列

### 样例输入

BADC

BDCA

### 样例输出

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  string in, post, ans = "";
5  void pre(int root, int l, int r) {
6      if (l > r) return;
7      int i = r;
8      while (post[root] == in[i]) i--;
9      ans += post[i];
10     pre(root - 1 + i - 1, l, i - 1);
11     pre(root - 1, i + 1, r);
12 }
13 int main() {
14     cin >> in >> post;
15     pre(post.length() - 1, 0, in.length() - 1);
16     cout << ans;
17     return 0;
18 }

```

## ALGO-21. 装箱问题（动态规划，01背包）

### 问题描述

有一个箱子容量为 $V$ （正整数， $0 \leq V \leq 20000$ ），同时有 $n$ 个物品（ $0 < n \leq 30$ ），每个物品有一个体积（正整数）。

要求 $n$ 个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

### 输入格式

第一行为一个整数，表示箱子容量；

第二行为一个整数，表示有 $n$ 个物品；

接下来 $n$ 行，每行一个整数表示这 $n$ 个物品的各自体积。

### 输出格式

一个整数，表示箱子剩余空间。

### 样例输入

24

6

8

3

12

7

9

7

样例输出

0

分析：dp[i][j]表示前i件物品选则部分装入体积为j的背包后，背包总共所占的最大体积，

一共有n件物品，那么dp[n][v]就是前n件物品选择部分装入体积为v的背包后，背包总共占有的最大体积

1.当当前输入的物品体积大于背包容量，则不装入背包，dp[i][j] = dp[i-1][j];

2.当当前输入的物品体积小于等于背包容量，考虑装或者不装两种状态，取体积最大的那个：dp[i][j] = max(dp[i-1][j], dp[i-1][j-t] + t);

```
1  #include <iostream>
2  using namespace std;
3  int dp[31][20001];
4  int main() {
5      int v, n;
6      cin >> v >> n;
7      for(int i = 1; i <= n; i++) {
8          int t;
9          cin >> t;
10         for(int j = 1; j <= v; j++) {
11             if (j >= t) {
12                 dp[i][j] = max(dp[i-1][j], dp[i-1][j-t] + t);
13             } else {
14                 dp[i][j] = dp[i-1][j];
15             }
16         }
17     }
18     cout << v - dp[n][v];
19     return 0;
20 }
```

## ALGO-22. 数的划分

问题描述

将整数n分成k份，且每份不能为空，任意两份不能相同(不考虑顺序)。

例如：n=7，k=3，下面三种分法被认为是相同的。

1, 1, 5; 1, 5, 1; 5, 1, 1;

问有多少种不同的分法。

输入格式

n, k

输出格式

一个整数，即不同的分法

样例输入

7 3

样例输出

4 {四种分法为：1，1，5;1，2，4;1，3，3;2，2，3;}

数据规模和约定

$6 < n \leq 200, 2 \leq k \leq 6$

分析：递归问题，step表示当前剩余的数需要分成的份数~~

把n分成k份，只需第一个数等于i，计算从i等于1一直到i等于n/k，然后把剩余的n-i分成k-1份的种类数...

front为剩余的要划分的数的前一个数，每次i从front开始一直到n/step结束，这样才能保证得到的划分方式是不递减的，才能保证不会有重复的情况产生~

```
1  #include <iostream>
2  using namespace std;
3  int cnt = 0;
4
5  void dfs(int front, int n, int step) {
6      if(step == 1) {
7          cnt++;
8          return ;
9      }
10     for(int i = front; i <= n / step; i++)
11         dfs(i, n - i, step - 1);
12 }
13
14 int main() {
15     int n, k;
16     cin >> n >> k;
17     dfs(1, n, k);
18     cout << cnt;
19     return 0;
20 }
```

## ALGO-23. 一元三次方程求解

问题描述

有形如： $ax^3+bx^2+cx+d=0$  这样的一个一元三次方程。给出该方程中各项的系数(a, b, c, d 均为实数)，并约定该方程存在三个不同实根(根的范围在-100至100之间)，且根与根之差的绝对值 $\geq 1$ 。要求三个实根。。



输入格式

四个实数：a, b, c, d

输出格式

由小到大依次在同一行输出这三个实根(根与根之间留有空格)，并精确到小数点后2位

样例输入

1 -5 -4 20

样例输出

-2.00 2.00 5.00

数据规模和约定

$|a|, |b|, |c|, |d| \leq 10$

分析：1.若  $x_1 < x_2$ ，且  $f(x_1) \cdot f(x_2) < 0$ ，则在  $(x_1, x_2)$  之间一定有一个根

2.找到一个区间内的根，二分逼近，找到误差很小的近似根，可认为是根

3.题目描述每个根间隔最少为1，从-100到100遍历，遇到整数根直接输出，否则判断是否在其右侧区间并查找~

```
1  #include <iostream>
2  using namespace std;
3  double a, b, c, d;
4  double f(double x) {
5      return a * x * x * x + b * x * x + c * x + d;
6  }
7  double find(double x, double y) {
8      if (y - x < 0.0001) return x;
9      double mid = (x + y) / 2;
10     if (f(mid) * f(x) > 0) return find(mid, y);
11     else return find(x, mid);
12 }
13 int main() {
14     cin >> a >> b >> c >> d;
15     for (int i = -100; i <= 100; i++) {
16         if (f(i) == 0) printf("%.2f ", i * 1.0);
17         else if (f(i) * f(i + 1) < 0) printf("%.2f ", find(i, i + 1));
18     }
19     return 0;
20 }
```

## ALGO-25. Car的旅行路线

问题描述

又到暑假了，住在城市A的Car想和朋友一起去城市B旅游。她知道每个城市都有四个飞机场，分别位于一个矩形的四个顶点上，同一个城市中两个机场之间有一条笔直的高速铁路，第*i*个城市中高速铁路的单位里程价格为*T<sub>i</sub>*，任意两个不同城市的机场之间均有航线，所有航线单位里程的价格均为*t*。那么Car应如何安排到城市B的路线才能尽可能的节省花费呢？她发现这并不是一个简单的问题，于是她来向你请教。

找出一条从城市A到B的旅游路线，出发和到达城市中的机场可以任意选取，要求总的花费最少。

输入格式

的第一行有四个正整数*s*, *t*, *A*, *B*。

*S*表示城市的个数，*t*表示飞机单位里程的价格，*A*, *B*分别为城市A, B的序号，( $1 \leq A, B \leq S$ )。

接下来有*S*行，其中第*i*行均有7个正整数*xi1*, *yi1*, *xi2*, *yi2*, *xi3*, *yi3*, *T<sub>i</sub>*，这当中的(*xi1*, *yi1*), (*xi2*, *yi2*), (*xi3*, *yi3*)分别是第*i*个城市中任意三个机场的坐标，*T<sub>i</sub>*为第*i*个城市高速铁路单位里程的价格。

输出格式

共有*n*行，每行一个数据对应测试数据，保留一位小数。

样例输入

```
1
1 10 1 3
1 1 1 3 3 1 30
2 5 7 4 5 2 1
8 6 8 8 11 6 3
```

样例输出

```
47.55
```

数据规模和约定

$0 < S \leq 100$ ,

分析：1.每个城市的四个机场相互有铁路联通，非同一城市的机场相互有航线，所以*n*个城市有4\*n个机场

2.给出三个矩形的定点，通过向量垂直判断直角点，再通过中点坐标公式计算出另外一个点，把四个机场点的相互到达所需花费记录起来

3.把每两个不同城市机场之间的花费记录起来

4.得到一个4*n*\*4*n*的邻接矩阵表示每个点到其他点的花费，用迪杰斯特拉算法循环扫描四次，{*s1,s2,s3,s4*}到{*e1,e2,e3,e4*}之间的最短路径，共16条中取最小即可~

```
1  #include <iostream>
2  #include <vector>
3  #include <cmath>
4  using namespace std;
5  struct node {
6      int x, y, id;
7  };
8  vector<node> nodes(1);
9  int n, flyprice, beginn, endn, inf = 99999;
```

```

10 double e[500][500];
11 double w(int x1, int y1, int x2, int y2) {
12     return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
13 }
14 bool h(int x1, int y1, int x2, int y2, int x3, int y3) {
15     return !((x1 - x2) * (x1 - x3) + (y1 - y2) * (y1 - y3));
16 }
17 double f(int m) {
18     double dis[500];
19     bool book[500] = {0};
20     fill(dis, dis + 500, inf);
21     for (int i = 0; i < 500; i++) dis[i] = inf;
22     dis[m] = 0;
23     for (int i = 1; i <= n * 4 - 1; i++) {
24         int minn = inf, u = -1;
25         for (int j = 1; j <= n * 4; j++) {
26             if (book[j] == false && dis[j] < minn) {
27                 minn = dis[j];
28                 u = j;
29             }
30         }
31         if (u == -1) break;
32         book[u] = true;
33         for (int k = 1; k <= n * 4; k++) {
34             if (e[u][k] < inf && dis[k] > e[u][k] + dis[u]) {
35                 dis[k] = e[u][k] + dis[u];
36             }
37         }
38     }
39     return min(min(dis[endn * 4 - 3], dis[endn * 4 - 2]), min(dis[endn
40 * 4 - 1], dis[endn * 4]));
41 }
42 int main() {
43     fill(e[0], e[0] + 500 * 500, inf);
44     scanf("%d %d %d %d", &n, &flyprice, &beginn, &endn);
45     for (int i = 1; i <= n; i++) {
46         int x1, y1, x2, y2, x3, y3, x4, y4, t;
47         scanf("%d %d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3,
48 &t);
49         if (h(x1, y1, x2, y2, x3, y3)) {
50             x4 = x2 + x3 - x1;
51             y4 = y2 + y3 - y1;
52         } else if (h(x2, y2, x1, y1, x3, y3)) {
53             x4 = x1 + x3 - x2;
54             y4 = y1 + y3 - y2;
55         } else {
56             x4 = x1 + x2 - x3;
57             y4 = y1 + y2 - y3;
58         }
59     }
60 }

```

```

57         e[(i - 1) * 4 + 1][(i - 1) * 4 + 2] = e[(i - 1) * 4 + 2][(i -
58         1) * 4 + 1] = w(x1, y1, x2, y2) * t;
59         e[(i - 1) * 4 + 1][(i - 1) * 4 + 3] = e[(i - 1) * 4 + 3][(i -
60         1) * 4 + 1] = w(x1, y1, x3, y3) * t;
61         e[(i - 1) * 4 + 1][(i - 1) * 4 + 4] = e[(i - 1) * 4 + 4][(i -
62         1) * 4 + 1] = w(x1, y1, x4, y4) * t;
63         e[(i - 1) * 4 + 2][(i - 1) * 4 + 3] = e[(i - 1) * 4 + 3][(i -
64         1) * 4 + 2] = w(x2, y2, x3, y3) * t;
65         e[(i - 1) * 4 + 2][(i - 1) * 4 + 4] = e[(i - 1) * 4 + 4][(i -
66         1) * 4 + 2] = w(x2, y2, x4, y4) * t;
67         e[(i - 1) * 4 + 3][(i - 1) * 4 + 4] = e[(i - 1) * 4 + 4][(i -
68         1) * 4 + 3] = w(x3, x3, x4, y4) * t;
69         nodes.push_back({x1, y1, i});
70         nodes.push_back({x2, y2, i});
71         nodes.push_back({x3, y3, i});
72         nodes.push_back({x4, y4, i});
73     }
74     for (int i = 1; i <= n * 4; i++) {
75         for (int j = i + 1; j <= n * 4; j++) {
76             if (nodes[i].id != nodes[j].id) {
77                 e[i][j] = e[j][i] = flyprice * w(nodes[i].x,
78                 nodes[i].y, nodes[j].x, nodes[j].y);
79             }
80         }
81     }
82     double ans = inf;
83     for (int i = 3; i >= 0; i--)
84         ans = min(ans, f(beginn * 4 - i));
85     printf("%.1f\n", ans);
86     return 0;
87 }

```

## ALGO-26. 麦森数

### 问题描述

形如 $2^P-1$ 的素数称为麦森数，这时 $P$ 一定也是个素数。但反过来不一定，即如果 $P$ 是个素数， $2^P-1$ 不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是 $P=3021377$ ，它有909526位。麦森数有许多重要应用，它与完全数密切相关。

任务：从文件中输入 $P$  ( $1000 < P < 3100000$ )，计算 $2^P-1$ 的位数和最后500位数字（用十进制高精度数表示）

### 输入格式

文件中只包含一个整数 $P$  ( $1000 < P < 3100000$ )

### 输出格式

不必验证 $2P-1$ 与 $P$ 是否为素数。

### 样例输入

1279

### 样例输出

386

[illegible]

分析：1.快速幂二分求 $2^p$ ，用字符串模拟乘法，注意，每次只保留后505数字即可（全部保留超时）

2.求位数直接公式  $\log_{10}(2^p)+1 = p*\log(2)+1$  即可

3.取后500位时, 直接在前面先加上500位0, 再取后500位, 这样可省去判断过程~

```

1  #include <iostream>
2  #include <algorithm>
3  #include <vector>
4  #include <string>
5  #include <cmath>
6  using namespace std;
7  string mul(string s1, string s2) {
8      if (s1.length() > 505) s1 = s1.substr(s1.length() - 505);
9      if (s2.length() > 505) s1 = s2.substr(s2.length() - 505);
10     vector<int> v(s1.length() + s2.length(), 0);
11     reverse(s1.begin(), s1.end());
12     reverse(s2.begin(), s2.end());
13     for (int i = 0; i < s1.length(); i++) {
14         for (int j = 0; j < s2.length(); j++) {
15             int t = (s1[i] - '0') * (s2[j] - '0') + v[i + j];
16             v[i + j] = t % 10;
17             v[i + j + 1] += t / 10;
18         }
19     }
20     int i;
21     for (i = v.size() - 1; v[i] == 0; i--);
22     string ans(i + 1, '0');
23     for (int j = 0; j <= i; j++)

```

```

24         ans[j] = v[i - j] + '0';
25     return ans;
26
27 }
28 string f(int p) {
29     if (p == 1) return "2";
30     if (p % 2 == 1) return mul(f(p - 1), "2");
31     string s = f(p / 2);
32     return mul(s, s);
33 }
34
35 string cut(string s, int index) {
36     if (s[index] != 0) {
37         s[index] = s[index] - 1;
38         return s;
39     } else {
40         s[index] = '9';
41         return cut(s, index - 1);
42     }
43 }
44 int main() {
45     int p;
46     cin >> p;
47     string s = f(p);
48     string ans = cut(s, s.length() - 1);
49     cout << (int) (p * log10(2)) + 1 << endl;
50     string add(500, '0');
51     ans = add + ans;
52     ans = ans.substr(ans.length() - 500);
53     for (int i = 0; i < 500; i++) {
54         cout << ans[i];
55         if ((i + 1) % 50 == 0) cout << endl;
56     }
57     return 0;
58 }

```

## ALGO-28. 星际交流

### 问题描述

人类终于登上了火星的土地并且见到了神秘的火星人。人类和火星人都无法理解对方的语言，但是我们的科学家发明了一种用数字交流的方法。这种交流方法是这样的，首先，火星人把一个非常大的数字告诉人类科学家，科学家破解这个数字的含义后，再把一个很小的数字加到这个大数上面，把结果告诉火星人，作为人类的回答。

火星人用一种非常简单的方式来表示数字——掰手指。火星人只有一只手，但这只手上有成千上万的手指，这些手指排成一列，分别编号为1，2，3……。火星人的任意两根手指都能随意交换位置，他们就是通过这方法计数的。

一个火星人用一个人类的手演示了如何用手指计数。如果把五根手指——拇指、食指、中指、无名指和小指分别编号为1，2，3，4和5，当它们按正常顺序排列时，形成了5位数12345，当你交换无名指

和小指的位置时，会形成5位数12354，当你把五个手指的顺序完全颠倒时，会形成54321，在所有能够形成的120个5位数中，12345最小，它表示1；12354第二小，它表示2；54321最大，它表示120。下表展示了只有3根手指时能够形成的6个3位数和它们代表的数字：

三进制数

123

132

213

231

312

321

代表的数字

1

2

3

4

5

6

现在你有幸成为了第一个和火星星人交流的地球人。一个火星星人会让你看他的手指，科学家会告诉你你要加上去的很小的数。你的任务是，把火星星人用手指表示的数与科学家告诉你的数相加，并根据相加的结果改变火星星人手指的排列顺序。输入数据保证这个结果不会超出火星星人手指能表示的范围。

输入格式

包括三行，第一行有一个正整数N，表示火星星人手指的数目（ $1 \leq N \leq 10000$ ）。第二行是一个正整数M，表示要加上去的小整数（ $1 \leq M \leq 100$ ）。下一行是1到N这N个整数的一个排列，用空格隔开，表示火星星人手指的排列顺序。

输出格式

只有一行，这一行含有N个整数，表示改变后的火星星人手指的排列顺序。每两个相邻的数中间用一个空格分开，不能有多余的空格。

样例输入

5

3

1 2 3 4 5

样例输出

1 2 4 5 3

数据规模和约定

对于30%的数据， $N \leq 15$ ；

对于60%的数据， $N \leq 50$ ；

对于全部的数据， $N \leq 10000$ ；

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4 int main() {
```

```

5      int n, m, a[10001];
6      cin >> n >> m;
7      for(int i = 0; i < n; i++)
8          cin >> a[i];
9      for(int i = 0; i < m; i++)
10         next_permutation(a,a+n);
11     for(int i = 0; i < n; i++)
12         cout << a[i] << " ";
13     return 0;
14 }

```

## ALGO-29. 校门外的树（区间处理）

### 问题描述

某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数轴上的每个整数点，即0，1，2，……，L，都种有一棵树。

由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树都移走后，马路上还有多少棵树。

### 输入格式

输入文件的第一行有两个整数L（ $1 \leq L \leq 10000$ ）和M（ $1 \leq M \leq 100$ ），L代表马路的长度，M代表区域的数目，L和M之间用一个空格隔开。接下来的M行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点和终止点的坐标。

### 输出格式

输出文件包括一行，这一行只包含一个整数，表示马路上剩余的树的数目。

### 样例输入

```

500 3
150 300
100 200
470 471

```

### 样例输出

```

298

```

### 数据规模和约定

对于20%的数据，区域之间没有重合的部分；

对于其它的数据，区域之间有重合的情况。

```

1  #include <iostream>
2  #include <vector>

```



```

3  using namespace std;
4  int main() {
5      int l, m;
6      cin >> l >> m;
7      vector<int> v(l+1, 1);
8      for(int i = 0; i < m; i++) {
9          int a, b;
10         cin >> a >> b;
11         for(int j = a; j <= b; j++) {
12             v[j] = 0;
13         }
14     }
15     int cnt = 0;
16     for(int i = 0; i < l+1; i++) {
17         if(v[i] == 1)
18             cnt++;
19     }
20     cout << cnt;
21     return 0;
22 }

```

## ALGO-30. 入学考试（01背包，动态规划）

### 问题描述

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

第一行有两个整数T（ $1 \leq T \leq 1000$ ）和M（ $1 \leq M \leq 100$ ），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100之间（包括1和100）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

包括一行，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。

### 样例输入

70 3

71 100

69 1

1 2

### 样例输出

## 数据规模和约定

对于30%的数据， $M \leq 10$ ；

对于全部的数据， $M \leq 100$ 。

分析：01背包问题。对于每一个输入都有采和不采两种状态。

$dp[i][j]$ 表示对于前*i*个草药选择部分采且总时间不超过*j*小时后，草药的价值的最大值

可得 $dp[M][T]$ 即是所求的解。

1.当当前输入的草药所需时间大于允许的最大时间*j*小时的时候，则不采， $dp[i][j] = dp[i-1][j]$ ;

2.当当前输入的草药所需时间小于等于允许的最大时间小时的时候，考虑采或者不采两种状态，取能够使草药总价值最大的那个： $dp[i][j] = \max(dp[i-1][j], dp[i-1][j-a] + b)$ ;

```

1  #include <iostream>
2  using namespace std;
3  int dp[101][1001];
4  int main() {
5      int t, m;
6      cin >> t >> m;
7      for(int i = 1; i <= m; i++) {
8          int a, b;
9          cin >> a >> b;
10         for(int j = 1; j <= t; j++) {
11             if(j >= a)
12                 dp[i][j] = max(dp[i-1][j], dp[i-1][j-a] + b);
13             else
14                 dp[i][j] = dp[i-1][j];
15         }
16     }
17     cout << dp[m][t];
18     return 0;
19 }
```

## ALGO-31. 开心的金明（01背包，动态规划）

### 问题描述

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：

“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过N元钱就行”。今天一早金明就开始做预算，但是他想买的东西太多了，

肯定会超过妈妈限定的N元。于是，他把每件物品规定了一个重要度，分为5等：用整数1~5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过N元（可以等于N元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

设第j件物品的价格为v[j]，重要度为w[j]，共选中了k件物品，编号依次为j1，j2，.....，jk，则所求的总和为：

$v[j1]*w[j1]+v[j2]*w[j2]+...+v[jk]*w[jk]$ 。（其中\*为乘号）

请 你帮助金明设计一个满足要求的购物单。

输入格式

输入文件 的第1行，为两个正整数，用一个空格隔开：

N m

（其中N（<30000）表示总钱 数，m（<25）为希望购买物品的个数。）

从第2行到第m+1行，第j行给出了编号为j-1的物品的的基本数据，每行有2个非负整数

v p

（其中v表示该物品的价格(v<=10000)，p表示该物品的重要度(1~5)）

输出格式

输出文件只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（<1000000000）。

样例输入

```
1000 5
800 2
400 5
300 5
400 3
200 2
```

样例输出

```
3900
```

分析：01背包问题。对于每一个输入都有买和不买两种状态。

dp[i][j]表示对于前i件物品选择部分购买限定总价不超过j元后，物品的价格与重要程度乘积的总和的最大值

可得dp[m][n]即是所求的解。

1.当当前输入的物品价格大于允许的最大总价j元，则不买，dp[i][j] = dp[i-1][j];

2.当当前输入的物品体积小于等于允许的最大总价j元，考虑买或者不买两种状态，取物品的价格与重要程度乘积的总和最大的那个：dp[i][j] = max(dp[i-1][j], dp[i-1][j-a] + t);

```
1  #include <iostream>
2  using namespace std;
3  int dp[25][30000];
4  int main() {
5      int n, m;
```

```

6      cin >> n >> m;
7      for(int i = 1; i <= m; i++) {
8          int a, b;
9          cin >> a >> b;
10         for(int j = 1; j <= n; j++) {
11             if(j >= a)
12                 dp[i][j] = max(dp[i-1][j], dp[i-1][j-a] + a * b);
13             else
14                 dp[i][j] = dp[i-1][j];
15         }
16     }
17     cout << dp[m][n];
18     return 0;
19 }

```

## ALGO-34. 纪念品分组（贪心算法+排序）

### 问题描述

元旦快到了，校学生会让乐乐负责新年晚会的纪念品发放工作。为使得参加晚会的同学所获得的纪念品价值相对均衡，他要把购来的纪念品根据价格进行分组，但每组最多只能包括两件纪念品，并且每组纪念品的价格之和不能超过一个给定的整数。为了保证在尽量短的时间内发完所有纪念品，乐乐希望分组的数目最少。

你的任务是写一个程序，找出所有分组方案中分组数最少的一种，输出最少的分组数目。

### 输入格式

输入包含 $n+2$ 行：

第1行包括一个整数 $w$ ，为每组纪念品价格之和的上限。

第2行为一个整数 $n$ ，表示购来的纪念品的总件数。

第3~ $n+2$ 行每行包含一个正整数 $p_i$  ( $5 \leq p_i \leq w$ )，表示所对应纪念品的价格。

### 输出格式

输出仅一行，包含一个整数，即最少的分组数目。

### 样例输入

```

100
9
90
20
20
30
50
60
70
80
90

```

样例输出

6

数据规模和约定

50%的数据满足：  $1 \leq n \leq 15$

100%的数据满足：  $1 \leq n \leq 30000, 80 \leq w \leq 200$

分析：排序+贪心。先从小到大排序，i、j指针分别从左到右、从右到左遍历。

如果  $a[i] + a[j] \leq w$ ，那么把这两个物品都放入同一组，并且同时移动指针；

否则，只能把j所指向的物品放入单独的一个组，移动j指针...直到j把所有物品都遍历完~

分析下贪心算法的可行性：如果j物品和i物品加起来超过了w，因为j物品比j+1处的物品价值小，那么如果i不能满足加起来的条件，而i-1能满足该条件，是否会影响贪心算法的结果呢？

如果让j和i-1去组合，对于j+1，因为价值比j更大，所以就更放不进i了，所以即使交换组合方式还是不影响构成的组数的~

至于为什么循环的条件是  $i \leq j$ ，当  $i < j$  的时候，假设ij已经相邻，那么此时还能再比较一次  $i+j$  能否满足构成一组的条件，如果满足就会把它们放到一组，之后ij指针同时移动后  $i > j$  不能够满足进入循环的条件了；如果不满足把它们放入一组，那么j放入一组后j移动指针，使  $i == j$ ，继续进入循环；

当  $i == j$  的时候，因为这个单独的物品已经没有可以组合的物品剩余了，所以此时还要进入循环进行一次  $\text{cnt}++$  的计数操作~//这就是  $\text{while}(i \leq j)$  的解释~~~

```
1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  using namespace std;
5  int main() {
6      int w, n;
7      scanf("%d %d", &w, &n);
8      int *a = new int[n];
9      for(int i = 0; i < n; i++)
10         scanf("%d", &a[i]);
11     sort(a, a+n);
12     int i = 0, j = n-1, cnt = 0;
13     while(i <= j) {
14         if(a[i] + a[j] <= w) {
15             i++;
16         }
17         cnt++;
18         j--;
19     }
20     cout << cnt;
21     return 0;
22 }
```

## ALGO-37. Hankson的趣味题

## 问题描述

Hanks 博士是BT (Bio-Tech, 生物技术) 领域的知名专家, 他的儿子名叫Hankson。现在, 刚刚放学回家的Hankson 正在思考一个有趣的问题。今天在课堂上, 老师讲解了如何求两个正整数 $c_1$  和 $c_2$  的最大公约数和最小公倍数。现在Hankson 认为自己已经熟练地掌握了这些知识, 他开始思考一个“求公约数”和“求公倍数”之类问题的“逆问题”, 这个问题是这样的: 已知正整数 $a_0, a_1, b_0, b_1$ , 设某未知正整数 $x$  满足: 1.  $x$  和 $a_0$  的最大公约数是 $a_1$ ; 2.  $x$  和 $b_0$  的最小公倍数是 $b_1$ 。Hankson 的“逆问题”就是求出满足条件的正整数 $x$ 。但稍加思索之后, 他发现这样的  $x$  并不唯一, 甚至可能不存在。因此他转而开始考虑如何求解满足条件的 $x$  的个数。请你帮助他编程求解这个问题。

## 输入格式

输入第一行为一个正整数 $n$ , 表示有 $n$  组输入数据。

接下来的 $n$  行每行一组输入数据, 为四个正整数 $a_0, a_1, b_0, b_1$ , 每两个整数之间用一个空格隔开。输入数据保证 $a_0$  能被 $a_1$  整除,  $b_1$  能被 $b_0$  整除。

## 输出格式

输出共 $n$  行。每组输入数据的输出结果占一行, 为一个整数。

对于每组数据: 若不存在这样的  $x$ , 请输出0; 若存在这样的  $x$ , 请输出满足条件的 $x$  的个数;

## 样例输入

```
2
41 1 96 288
95 1 37 1776
```

## 样例输出

```
6
2
```

## 样例说明

第一组输入数据,  $x$  可以是9、18、36、72、144、288, 共有6 个。

第二组输入数据,  $x$  可以是48、1776, 共有2 个。

## 数据规模和约定

对于 50%的数据, 保证有 $1 \leq a_0, a_1, b_0, b_1 \leq 10000$  且 $n \leq 100$ 。

对于 100%的数据, 保证有 $1 \leq a_0, a_1, b_0, b_1 \leq 2,000,000,000$  且 $n \leq 2000$ 。

分析: 暴力枚举, 判断计数即可, 注意, 因子是成对出现的, 枚举到 $\sqrt{b_1}$ (即可),  $x = \sqrt{b_1}$ 只要判断一次即可~

```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 #include <string>
5 #include <cmath>
6 int gcd(int a, int b) {
7     if (b == 0) return a;
8     return gcd(b, a % b);
9 }
```

```

10  using namespace std;
11  int main() {
12      int a0, a1, b0, b1, k;
13      cin >> k;
14      while (k--) {
15          int ans = 0;
16          scanf("%d %d %d %d", &a0, &a1, &b0, &b1);
17          for (int i = 1; i * i <= b1; i++) {
18              if(b1 % i == 0){
19                  int n = i;
20                  if (gcd(a0, n) == a1 && b0 * n == b1 * gcd(b0, n))
21                      ans++;
22                  if(i != b1 / i) {
23                      n = b1 / i;
24                      if (gcd(a0, n) == a1 && b0 * n == b1 * gcd(b0, n))
25                          ans++;
26                  }
27              }
28          }
29          printf("%d\n",ans);
30      }
31      return 0;
32  }

```

## ALGO-38. 接水问题

### 问题描述

学校里有一个水房，水房里一共装有 $m$ 个龙头可供同学们打开水，每个龙头每秒钟的供水量相等，均为1。现在有 $n$ 名同学准备接水，他们的初始接水顺序已经确定。将这些同学按接水顺序从1到 $n$ 编号， $i$ 号同学的接水量为 $w_i$ 。接水开始时，1到 $m$ 号同学各占一个水龙头，并同时打开水龙头接水。当其中某名同学 $j$ 完成其接水量要求 $w_j$ 后，下一名排队等候接水的同学 $k$ 马上接替 $j$ 同学的位置开始接水。这个换人的过程是瞬间完成的，且没有任何水的浪费。即 $j$ 同学第 $x$ 秒结束时完成接水，则 $k$ 同学第 $x+1$ 秒立刻开始接水。若当前接水人数 $n'$ 不足 $m$ ，则只有 $n'$ 个龙头供水，其它 $m-n'$ 个龙头关闭。现在给出 $n$ 名同学的接水量，按照上述接水规则，问所有同学都接完水需要多少秒。

### 输入格式

第1行2个整数 $n$ 和 $m$ ，用一个空格隔开，分别表示接水人数和龙头个数。第2行 $n$ 个整数 $w_1$ 、 $w_2$ 、……、 $w_n$ ，每两个整数之间用一个空格隔开， $w_i$ 表示 $i$ 号同学的接水量。

### 输出格式

输出只有一行，1个整数，表示接水所需的总时间。

### 样例输入

```

5 3
4 4 1 2 1

```

### 样例输出

4

样例输入

8 4

23 71 87 32 70 93 80 76

样例输出

163

输入输出样例 1 说明

第1 秒，3 人接水。第1 秒结束时，1、2、3 号同学每人的已接水量为1，3 号同学接完水，4 号同学接替3 号同学开始接水。

第2 秒，3 人接水。第2 秒结束时，1、2 号同学每人的已接水量为2，4 号同学的已接水量为1。

第3 秒，3 人接水。第3 秒结束时，1、2 号同学每人的已接水量为3，4 号同学的已接水量为2。4 号同学接完水，5 号同学接替4 号同学开始接水。

第4 秒，3 人接水。第4 秒结束时，1、2 号同学每人的已接水量为4，5 号同学的已接水量为1。1、2、5 号同学接完水，即所有人完成接水。

总接水时间为4 秒。

数据规模和约定

$1 \leq n \leq 10000$ ,  $1 \leq m \leq 100$  且  $m \leq n$ ;

$1 \leq w_i \leq 100$ 。

分析：1.每次同学接水，都走到需要排队时间最少的队伍前，每次有同学排队，需最少排队时间的队都在变化

2.用优先队列，每次取出队头，把要进来的同学的排队时间加上，然后丢到队尾

3.最后，队伍最长的，就是接水总时间~

```
1  #include <iostream>
2  #include <vector>
3  #include <queue>
4  using namespace std;
5  int main() {
6      priority_queue<int, vector<int>, greater<int> > q;
7      int n, m, ans;
8      cin >> n >> m;
9      for (int i = 1; i <= n; i++) {
10         int t;
11         scanf("%d", &t);
12         if (i <= m) q.push(t);
13         else {
14             int a = q.top();
15             q.pop();
16             q.push(a + t);
17         }
18     }
19     while (!q.empty()) {
```



```

20         ans = q.top();
21         q.pop();
22     }
23     cout << ans;
24     return 0;
25 }

```

## ALGO-39. 数组排序去重

### 问题描述

输入10个整数组成的序列，要求对其进行升序排序，并去掉重复元素。

### 输入格式

10个整数。

### 输出格式

多行输出，每行一个元素。

### 样例输入

2 2 3 3 1 1 5 5 5 5

### 样例输出

1  
2  
3  
5

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int main() {
5      int a[10];
6      for(int i = 0; i < 10; i++) {
7          cin >> a[i];
8      }
9      sort(a, a+10);
10     cout << a[0] << endl;
11     for(int i = 1; i < 10; i++) {
12         if(a[i] != a[i-1]) {
13             cout << a[i] << endl;
14         }
15     }
16     return 0;
17 }

```

## ALGO-42. 送分啦

### 问题描述

这题想得分吗？想，请输出“yes”；不想，请输出“no”。

### 输出格式

输出包括一行，为“yes”或“no”。

分析：智障题目。。。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      cout << "yes";
5      return 0;
6  }
```

## ALGO-43. A+B Problem

### 问题描述

输入A,B。

输出A+B。

### 输入格式

输入包含两个整数A,B，用一个空格分隔。

### 输出格式

输出一个整数，表示A+B的值。

### 样例输入

5 8

### 样例输出

13

### 数据规模和约定

$-1,000,000,000 \leq A, B \leq 1,000,000,000$ 。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int A, B;
5      cin >> A >> B;
6      cout << A + B;
7      return 0;
8  }
```

## ALGO-45. 调和数列问题

### 问题描述

输入一个实数x，求最小的n使得， $1/2+1/3+1/4+\dots+1/(n+1)\geq x$ 。

输入的实数x保证大于等于0.01，小于等于5.20，并且恰好有两位小数。你的程序要能够处理多组数据，即不停地读入x，如果x不等于0.00，则计算答案，否则退出程序。

输出格式为对于一个x，输出一行n card(s)。其中n表示要计算的答案。

### 输入格式

分行输入x的具体数值

### 输出格式

分行输出n的数值，格式为n card(s)

### 样例输入

1.00

3.71

0.04

5.19

0.00

### 样例输入

3 card(s)

61 card(s)

1 card(s)

273 card(s)

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      double x, ans = 0;
5      cin >> x;
6      if(x != 0){
7          for(int i = 1; ; i++){
8              ans += 1.0 / (i+1);
9              if(ans >= x){
10                 printf("%d card(s)\n", i);
11                 return main();
12             }
13         }
14     } else{
15         return 0;
16     }
17 }
```

## ALGO-46. Hanoi问题

### 问题描述

如果将课本上的Hanoi塔问题稍做修改：仍然是给定N只盘子，3根柱子，但是允许每次最多移动相邻的M只盘子（当然移动盘子的数目也可以小于M），最少需要多少次？

例如N=5，M=2时，可以分别将最小的2个盘子、中间的2个盘子以及最大的一个盘子分别看作一个整体，这样可以转变为N=3，M=1的情况，共需要移动7次。

### 输入格式

输入数据仅有一行，包括两个数N和M（ $0 \leq M \leq N \leq 8$ ）

### 输出格式

仅输出一个数，表示需要移动的最少次数

### 样例输入

5 2

### 样例输出

7

分析：数学问题，递归得出公式 $f(n) = 2f(n-1) + 1$ ，然后通过数学推算得通项公式为 $f(n) = 2^n - 1$

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int n, m, ans = 0;
6      cin >> n >> m;
7      if (n % m == 0) n /= m;
8      else n = n / m + 1;
9      cout << (int) pow(2, n) - 1;
10     return 0;
11 }
```

## ALGO-47. 蜜蜂飞舞

### 问题描述

“两只小蜜蜂呀，飞在花丛中呀……”

话说这天天上飞舞着两只蜜蜂，它们在跳一种奇怪的舞蹈。用一个空间直角坐标系来描述这个世界，那么这两只蜜蜂初始坐标分别为 $(x_1, y_1, z_1)$ ， $(x_2, y_2, z_2)$ 。在接下来它们将进行n次飞行，第i次飞行两只蜜蜂分别按照各自的速度向量飞行 $t_i$ 个单位时间。对于这一现象，玮玮已经观察了很久。他很想知道在蜜蜂飞舞结束时，两只蜜蜂的距离是多少。现在他就求教于你，请你写一个程序来帮他计算这个结果。

### 输入格式

第一行有且仅有一个整数 $n$ ，表示两只蜜蜂将进行 $n$ 次飞行。

接下来有 $n$ 行。

第 $i$ 行有7个用空格分隔开的整数 $a_i, b_i, c_i, d_i, e_i, f_i, t_i$ ，表示第一只蜜蜂单位时间的速度向量为 $(a_i, b_i, c_i)$ ，第二只蜜蜂单位时间的速度向量为 $(d_i, e_i, f_i)$ ，它们飞行的时间为 $t_i$ 。

最后一行有6个用空格分隔开的整数 $x_1, y_1, z_1, x_2, y_2, z_2$ ，如题所示表示两只蜜蜂的初始坐标。

输出格式

输出仅包含一行，表示最后两只蜜蜂之间的距离。保留4位小数位。

样例输入

Sample 1

```
1
1 1 1 1 -1 1 2
3 0 1 2 0 0
```

Sample 2

```
3
1 1 1 1 -1 1 2
2 1 2 0 -1 -1 2
2 0 0 -1 1 1 3
3 0 1 2 0 0
```

样例输出

Sample 1

4.2426

Sample 2

15.3948

分析：1.全排列公式，每次排列后检查即可检查

2.题目说的后面，是紧跟着后面，中间隔着一个在后面不算的~

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int n, x1 = 0, y1 = 0, z1 = 0, x2 = 0, y2 = 0, z2 = 0;
6      cin >> n;
7      for (int i = 0; i <= n; i++) {
8          int tx1, ty1, tz1, tx2, ty2, tz2, t;
9          cin >> tx1 >> ty1 >> tz1 >> tx2 >> ty2 >> tz2;
10         if (i == n) t = 1;
11         else cin >> t;
12         x1 += tx1 * t;
13         y1 += ty1 * t;
14         z1 += tz1 * t;
15         x2 += tx2 * t;
16         y2 += ty2 * t;
17         z2 += tz2 * t;
18     }
```

```

19     double ans = sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) +
20         (z2 - z1) * (z2 - z1));
21     printf("%.4f", ans);
22     return 0;
23 }

```

## ALGO-48. 关联矩阵

### 问题描述

有一个n个结点m条边的有向图，请输出他的关联矩阵。

### 输入格式

第一行两个整数n、m，表示图中结点和边的数目。n<=100,m<=1000。

接下来m行，每行两个整数a、b，表示图中有(a,b)边。

注意图中可能含有重边，但不会有自环。

### 输出格式

输出该图的关联矩阵，注意请勿改变边和结点的顺序。

### 样例输入

```

5 9
1 2
3 1
1 5
2 5
2 3
2 3
3 2
4 3
5 4

```

### 样例输出

```

1 -1 1 0 0 0 0 0 0
-1 0 0 1 1 1 -1 0 0
0 1 0 0 -1 -1 1 -1 0
0 0 0 0 0 0 0 1 -1
0 0 -1 -1 0 0 0 0 1

```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4 int main() {
5     int n, m;
6     cin >> n >> m;
7     vector<vector<int>> > t(n, vector<int>(m));
8     for (int i = 0; i < m; i++) {

```

```

9         int a, b;
10        cin >> a >> b;
11        t[a - 1][i] = 1;
12        t[b - 1][i] = -1;
13    }
14    for (int i = 0; i < n; i++) {
15        for (int j = 0; j < m; j++) {
16            cout << t[i][j] << " ";
17        }
18        cout << endl;
19    }
20    return 0;
21 }

```

## ALGO-49. 寻找数组中最大值

### 问题描述

对于给定整数数组a[],寻找其中最大值，并返回下标。

### 输入格式

整数数组a[],数组元素个数小于1等于100。输出数据分作两行：

第一行只有一个数，表示数组元素个数；第二行为数组的各个元素。

### 输出格式

输出最大值，及其下标

### 样例输入

3

3 2 1

### 样例输出

3 0

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int *arr = new int [n];
7      for (int i = 0; i < n; i++) {
8          cin >> arr[i];
9      }
10     int max = arr[0], t = 0;
11     for (int i = 1; i < n; i++) {
12         if (max < arr[i]) {
13             max = arr[i];

```

```

14         t = i;
15     }
16 }
17 cout << max << " " << t;
18 delete [] arr;
19 return 0;
20 }

```

## ALGO-50. 数组查找及替换

### 问题描述

给定某整数数组和某一整数b。要求删除数组中可以被b整除的所有元素，同时将该数组各元素按从小到大排序。如果数组元素数值在A到Z的ASCII之间，替换为对应字母。元素个数不超过100，b在1至100之间。

### 输入格式

第一行为数组元素个数和整数b

第二行为数组各个元素

### 输出格式

按照要求输出

### 样例输入

7 2

77 11 66 22 44 33 55

### 样例输出

11 33 55 M

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int main() {
5      int n, b;
6      cin >> n >> b;
7      vector<int> v;
8      for(int i = 0; i < n; i++) {
9          int temp;
10         cin >> temp;
11         if(temp % b != 0) {
12             v.push_back(temp);
13         }
14     }
15     sort(v.begin(), v.end());
16     for(int i = 0; i < v.size(); i++) {
17         if(v[i] >= 'A' && v[i] <= 'Z') {

```



```

18         cout << (char)(v[i]) << " ";
19     } else {
20         cout << v[i];
21     }
22 }
23 return 0;
24 }

```

## ALGO-51. Torry的困惑(基本型)[前n个质数的乘积]

### 问题描述

Torry从小喜爱数学。一天，老师告诉他，像2、3、5、7.....这样的数叫做质数。

Torry突然想到一个问题，前10、100、1000、10000.....个质数的乘积是多少呢？

于是Torry求助于会编程的你，请你算出前n个质数的乘积。不过，考虑到你才接触编程不久，

Torry只要你算出这个数模上50000的值。

### 输入格式

仅包含一个正整数n，其中n<=100000。

### 输出格式

输出一行，即前n个质数的乘积模50000的值。

### 样例输入

1

### 样例输出

2

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int count = 0;
7      int result = 2;
8      int temp = 3;
9      while ((count < (n - 1)) && n != 1) {
10         int i = 2;
11         for (i = 2; i < temp; i++) {
12             if (temp % i == 0) {
13                 break;
14             }
15         }
16         if (i == temp) {
17             result = result * temp;
18             result = result % 50000;

```

```

19         count++;
20     }
21     temp++;
22 }
23 if (n == 0){
24     cout << 0;
25 }else{
26     cout << result;
27 }
28 return 0;
29 }

```

## ALGO-52. 排列问题

### 问题描述

求一个0~N-1的排列（即每个数只能出现一次），给出限制条件（一张N\*N的表，第i行第j列的1或0，表示为j-1这个数不能出现在i-1这个数后面，并保证第i行第i列为0），将这个排列看成一个自然数，求从小到大排序第K个排列。

### 数据规模和约定

$N \leq 10$ ,  $K \leq 500000$

### 输入格式

第一行为N和K,接下来的N行，每行N个数，0表示不能，1表示能

### 输出格式

### 所求的排列

### 样例输入

```

3 2
0 1 1
1 0 0
0 1 0

```

### 样例输出

```

1 0 2

```

### 解释：

对于N=3的没有任何限制的情况

```

第一：0 1 2
第二：0 2 1
第三：1 0 2
第四：1 2 0
第五：2 0 1
第六：2 1 0

```

根据题目所给的限制条件由于2不能出现在1后面，0不能出现在2后面

```

第一：0 2 1
第二：1 0 2

```

```

1  #include <iostream>
2  #include <cmath>
3  #include <vector>
4  #include <algorithm>
5  using namespace std;
6  int main() {
7      int n, k, t, a[10], index[10], cnt = 1;
8      vector<pair<int, int> > v;
9      cin >> n >> k;
10     for (int i = 0; i < n; i++)
11         a[i] = i;
12     for (int i = 0; i < n; i++) {
13         for (int j = 0; j < n; j++) {
14             cin >> t;
15             if (t == 0 && i != j) {
16                 v.push_back({i, j});
17             }
18         }
19     }
20     do {
21         int flag = 0;
22         for (int i = 0; i < n; i++)
23             index[a[i]] = i;
24         for (int i = 0; i < v.size(); i++) {
25             if (index[v[i].second] - index[v[i].first] == 1) {
26                 flag = 1;
27                 break;
28             }
29         }
30         if (flag == 0) {
31             if (cnt == k) {
32                 for (int i = 0; i < n; i++)
33                     cout << a[i] << ' ';
34                 break;
35             } else {
36                 cnt++;
37             }
38         }
39     } while (next_permutation(a, a + n));
40     return 0;
41 }

```

### ALGO-53. 最小乘积(基本型)

#### 问题描述

给两组数，各n个。

请调整每组数的排列顺序，使得两组数据相同下标元素对应相乘，然后相加的和最小。要求程序输出这个最小值。

例如两组数分别为:1 3 -5和-2 4 1

那么对应乘积取和的最小值应为：

$$(-5) * 4 + 3 * (-2) + 1 * 1 = -25$$

输入格式

第一个行一个数T表示数据组数。后面每组数据，先读入一个n，接下来两行每行n个数，每个数的绝对值小于等于1000。

$$n \leq 8, T \leq 1000$$

输出格式

一个数表示答案。

样例输入

2

3

1 3 -5

-2 4 1

5

1 2 3 4 5

1 0 1 0 1

样例输出

-25

6

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  int cmp1(int a, int b){return a < b;}
6  int cmp2(int a, int b){return a > b;}
7
8  int sum1(int a[], int b[],int n) {
9      int sum = 0;
10     for (int i = 0; i < n; i++){
11         sum = sum + a[i] * b[i];
12     }
13     return sum;
```

```

14 }
15
16 int main() {
17     int T;
18     cin >> T;
19     int *sum = new int [T];
20     for (int i = 0; i < T; i++) {
21         int n;
22         cin >> n;
23         int *a = new int [n];
24         int *b = new int [n];
25         for (int j = 0; j < n; j++) {
26             cin >> a[j];
27         }
28         for (int j = 0; j < n; j++) {
29             cin >> b[j];
30         }
31         sort(a, a + n, cmp1);
32         sort(b, b + n, cmp2);
33         sum[i] = sum1(a, b, n);
34     }
35     for (int i = 0; i < T; i++) {
36         cout << sum[i] << endl;
37     }
38     delete [] sum;
39     return 0;
40 }

```

## ALGO-54. 简单加法(基本型)

### 问题描述

首先给出简单加法算式的定义：

如果有一个算式 $(i)+(i+1)+(i+2)$ , ( $i \geq 0$ ), 在计算的过程中, 没有任何一个数位出现了进位, 则称其为简单的加法算式。

例如:  $i=3$ 时,  $3+4+5=12$ , 有一个进位, 因此 $3+4+5$ 不是一个简单的加法算式; 又如 $i=112$ 时,  $112+113+114=339$ , 没有在任意数位上产生进位, 故 $112+113+114$ 是一个简单的加法算式。

问题: 给定一个正整数 $n$ , 问当 $i$ 大于等于0且小于 $n$ 时, 有多少个算式 $(i)+(i+1)+(i+2)$ 是简单加法算式。其中 $n < 10000$ 。

### 输入格式

一个整数, 表示 $n$

### 输出格式

一个整数, 表示简单加法算式的个数

### 样例输入

4

样例输出

3

分析：判断位数取以10为底的对数更快更方便～

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int n, ans = 1;
6      cin >> n;
7      for (int i = 0; i < n; i++)
8          if ((int) log10(3 * i + 3) == (int) log10(i))
9              ans++;
10     cout << ans;
11     return 0;
12 }
```

## ALGO-55. 矩阵加法

问题描述

给定两个 $N \times M$ 的矩阵，计算其和。其中：

$N$ 和 $M$ 大于等于1且小于等于100，矩阵元素的绝对值不超过1000。

输入格式

输入数据的第一行包含两个整数 $N$ 、 $M$ ，表示需要相加的两个矩阵的行数和列数。接下来 $2 \times N$ 行每行包含 $M$ 个数，其中前 $N$ 行表示第一个矩阵，后 $N$ 行表示第二个矩阵。

输出格式

你的程序需要输出一个 $N \times M$ 的矩阵，表示两个矩阵相加的结果。注意，输出中每行的最后不应有多余的空格，否则你的程序有可能被系统认为是Presentation Error

样例输入

2 2

1 2

3 4

5 6

7 8

样例输出

6 8

10 12

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, m, t, a[150][150];
```

```

5      cin >> n >> m;
6      for(int i = 0; i < n; i++){
7          for(int j = 0; j < m; j++) {
8              cin >> a[i][j];
9
10         }
11     }
12     for(int i = 0; i < n; i++){
13         for(int j = 0; j < m; j++) {
14             cin >> t;
15             a[i][j] += t;
16
17         }
18     }
19     for(int i = 0; i < n; i++){
20         for(int j = 0; j < m; j++) {
21             if(j != m - 1) cout << a[i][j] << ' ';
22             else cout << a[i][j] << endl;
23
24         }
25     }
26     return 0;
27 }

```

## ALGO-58. 字符串逆序

### 问题描述

给定一个字符串，将这个串的所有字母逆序后输出。

### 输入格式

输入包含一个字符串，长度不超过100，字符串中不含空格。

### 输出格式

输出包含一个字符串，为上面字符串的逆序。

### 样例输入

tsinsen

### 样例输出

nesnist

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s;
5      cin >> s;
6      for(int i = s.length() - 1; i >= 0; i--) {
7          cout << s[i];
8      }
9      return 0;
10 }

```

## ALGO-60. 矩阵乘方

### 问题描述

给定一个矩阵A,一个非负整数b和一个正整数m, 求A的b次方除m的余数。

其中一个 $n \times n$ 的矩阵除m的余数得到的仍是一个 $n \times n$ 的矩阵, 这个矩阵的每一个元素是原矩阵对应位置上的数除m的余数。

要计算这个问题, 可以将A连乘b次, 每次都对m求余, 但这种方法特别慢, 当b较大时无法使用。下面给出一种较快的算法(用 $A^b$ 表示A的b次方):

若 $b=0$ , 则 $A^b \% m = I \% m$ 。其中I表示单位矩阵。

若b为偶数, 则 $A^b \% m = (A^{b/2} \% m)^2 \% m$ , 即先把A乘b/2次方对m求余, 然后再平方后对m求余。

若b为奇数, 则 $A^b \% m = (A^{b-1} \% m) * a \% m$ , 即先求A乘b-1次方对m求余, 然后再乘A后对m求余。

这种方法速度较快, 请使用这种方法计算 $A^b \% m$ , 其中A是一个 $2 \times 2$ 的矩阵, m不大于10000。

### 输入格式

输入第一行包含两个整数b, m, 第二行和第三行每行两个整数, 为矩阵A。

### 输出格式

输出两行, 每行两个整数, 表示 $A^b \% m$ 的值。

### 样例输入

```

2 2
1 1
0 1

```

### 样例输出

```

1 0
0 1

```

分析: 1.按照题目要求即可, 和快速幂算法类似

2.注意 当相乘次数为0时, 输出单位矩阵 (题目要求), (但是后台测试数据给的是0 0 0 0!)

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  vector<int> f(vector<int> v, int k, int m) {
5      vector<int> ans(5);

```



```

6     if (k == 0) {
7         ans[1] = ans[4] = 1;
8         ans[2] = ans[3] = 0;
9     } else {
10        if (k % 2 == 0) {
11            vector<int> t = f(v, k / 2, m);
12            ans[1] = (t[1] * t[1] + t[2] * t[3]) % m;
13            ans[2] = (t[1] * t[2] + t[2] * t[4]) % m;
14            ans[3] = (t[3] * t[1] + t[4] * t[3]) % m;
15            ans[4] = (t[3] * t[2] + t[4] * t[4]) % m;
16        } else {
17            vector<int> t = f(v, k - 1, m);
18            ans[1] = (t[1] * v[1] + t[2] * v[3]) % m;
19            ans[2] = (t[1] * v[2] + t[2] * v[4]) % m;
20            ans[3] = (t[3] * v[1] + t[4] * v[3]) % m;
21            ans[4] = (t[3] * v[2] + t[4] * v[4]) % m;
22        }
23    }
24    return ans;
25 }
26 int main() {
27     int k, m, a, b, c, d;
28     vector<int> v(5), ans;
29     cin >> k >> m >> v[1] >> v[2] >> v[3] >> v[4];
30     ans = f(v, k, m);
31     printf("%d %d\n%d %d\n", ans[1], ans[2], ans[3], ans[4]);
32     return 0;
33 }

```

## ALGO-61. 奇偶判断

### 问题描述

能被2整除的数称为偶数，不能被2整除的数称为奇数。给一个整数x，判断x是奇数还是偶数。

### 输入格式

输入包括一个整数x， $0 \leq x \leq 100000000$ 。

### 输出格式

如果x是奇数，则输出“odd”，如果是偶数，则输出“even”。

### 样例输入

10

### 样例输出

even

### 样例输入

2009

样例输出

odd

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a;
5      cin >> a;
6      if(a % 2 == 0) {
7          cout << "even";
8      } else {
9          cout << "odd";
10     }
11     return 0;
12 }
```

## ALGO-62. 平方计算

问题描述

输入正整数 $a, m$ ，输出 $a^2 \% m$ ，其中 $^$ 表示乘方，即 $a^2$ 表示 $a$ 的平方， $\%$ 表示取余。

输入格式

输入包含两个整数 $a, m$ ， $a$ 不超过10000。

输出格式

输出一个整数，即 $a^2 \% m$ 的值。

样例输入

5 6

样例输出

1

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, m;
5      cin >> a >> m;
6      cout << (a * a) % m << endl;
7      return 0;
8  }
```

## ALGO-63. 乘法表

问题描述

输出九九乘法表。

输出格式

输出格式见下面的样例。乘号用“\*”表示。

样例输出

下面给出输出的前几行：

1\*1=1

2\*1=2 2\*2=4

3\*1=3 3\*2=6 3\*3=9

4\*1=4 4\*2=8 4\*3=12 4\*4=16

.....

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      for(int i = 1; i <= 9; i++) {
5          for(int j = 1; j <= i; j++) {
6              cout << i << "*" << j << "=" << i * j << " ";
7          }
8          cout << endl;
9      }
10     return 0;
11 }
```

## ALGO-64. 大小写判断

问题描述

给定一个英文字母判断这个字母是大写还是小写。

输入格式

输入只包含一个英文字母c。

输出格式

如果c是大写字母，输出“upper”，否则输出“lower”。

样例输入

x

样例输出

lower

样例输入

B

样例输出

upper

```
1 #include <iostream>
2 #include <cctype>
3 using namespace std;
4 int main() {
5     char c;
6     cin >> c;
7     if(isupper(c)) {
8         cout << "upper";
9     } else {
10        cout << "lower";
11    }
12    return 0;
13 }
```

## ALGO-65. 比赛安排

问题描述

设有有 $2n$  ( $n \leq 6$ ) 个球队进行单循环比赛，计划在 $2n - 1$ 天内完成，每个队每天进行一场比赛。设计一个比赛的安排，使在 $2n - 1$ 天内每个队都与不同的对手比赛。

输入格式

输入文件matchplan.in共一行，输入 $n$ 的数值。

输出格式

输出文件matchplan.out共  $(2n - 1)$  行，第 $i$ 行输出第 $i$ 天的比赛安排。

格式为：< $i$ > A-B, C-D, ..... 其中 $i$ 是天数，A, B分别为比赛双方的编号，每行共 $2n-1$ 个比赛场次。

样例输入

2

样例输出

<1>1-2,3-4

<2>1-3,2-4

<3>1-4,2-3

分析：1.根据题意，队伍序号小的的先安排先比赛，每个队伍每天都要比赛

2.每个队伍向后挑选最近队伍比赛，标记今天比过赛了，并标记某两个队伍比过赛~

```
1 #include <iostream>
2 #include <cstring>
3 #include <cmath>
4 using namespace std;
5 int main() {
6     int n, m, a[1000], mp[10000] = {0};
7     cin >> n;
```

```

8      m = pow(2, n);
9      for (int i = 1; i <= m - 1; i++) {
10         memset(a, 0, sizeof(a));
11         printf("<%d>", i);
12         for (int j = 1; j <= m; j++) {
13             if (a[j] == 0) {
14                 for (int k = j + 1; k <= m; k++) {
15                     if (mp[j * 100 + k] == 0 && a[k] == 0) {
16                         printf("%d-%d ", j, k);
17                         mp[j * 100 + k] = 1;
18                         a[j] = a[k] = 1;
19                         break;
20                     }
21                 }
22             }
23         }
24     }
25     cout << endl;
26 }
27 return 0;
28 }

```

## ALGO-66. 字符串编辑

### 问题描述

从键盘输入一个字符串（长度<=40个字符），并以字符‘.’结束。编辑功能有：

1 D：删除一个字符，命令的方式为：D a 其中a为被删除的字符，例如：D s 表示删除字符‘s’，若字符串中有多个‘s’，则删除第一次出现的。

2 I：插入一个字符，命令的格式为：I a1 a2 其中a1表示插入到指定字符前面，a2表示将要插入的字符。例如：I s d 表示在指定字符‘s’的前面插入字符‘d’，若原串中有多个‘s’，则插入在最后一个字符的前面。

3 R：替换一个字符，命令格式为：R a1 a2 其中a1为被替换的字符，a2为替换的字符，若在原串中有多个a1则应全部替换。

在编辑过程中，若出现被改的字符不存在时，则给出提示信息。

### 输入格式

输入共两行，第一行为原串(以‘.’结束)，第二行为命令（输入方式参见“问题描述”）。

### 输出格式

输出共一行，为修改后的字符串或输出指定字符不存在的提示信息。

### 样例输入

This is a book.

D s

### 样例输出

Thi is a book.

## 输入输出样例解释

命令为删去s，第一个在字符中出现的s在This中，即得到结果。

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string s, ans;
6      char c, ch, ch1, ch2;
7      getline(cin, s);
8      cin >> c;
9      if (c == 'D') {
10         cin >> ch;
11         int i;
12         for (i = 0; i < s.length(); i++) {
13             if (s[i] == ch) {
14                 i++;
15                 break;
16             } else {
17                 ans += s[i];
18             }
19         }
20         for (; i < s.length(); i++)
21             ans += s[i];
22     } else if (c == 'I') {
23         cin >> ch1 >> ch2;
24         int i;
25         for (i = s.length() - 1; i >= 0; i--) {
26             ans = s[i] + ans;
27             if (s[i] == ch1) {
28                 ans = ch2 + ans;
29                 i--;
30                 break;
31             }
32         }
33         for (; i >= 0; i--)
34             ans = s[i] + ans;
35     } else {
36         cin >> ch1 >> ch2;
37         for (int i = 0; i < s.length(); i++) {
38             if (s[i] == ch1) s[i] = ch2;
39             ans += s[i];
40         }
41     }
42     cout << ans << endl;
43     return 0;
44 }
```

## ALGO-67. 最大值与最小值的计算

输入11个整数，计算它们的最大值和最小值。

样例输入

0 1 2 3 4 5 6 7 8 9 10

样例输出

10 0

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int mmax = INT_MIN;
5      int mmin = INT_MAX;
6      for(int i = 0; i < 11; i++) {
7          int temp;
8          cin >> temp;
9          mmax = mmax > temp ? mmax : temp;
10         mmin = mmin < temp ? mmin : temp;
11     }
12     cout << mmax << " " << mmin;
13     return 0;
14 }
```

## ALGO-68. 判定数字

编写函数，判断某个给定字符是否为数字。

样例输入

9

样例输出

yes

```
1  #include <iostream>
2  #include <cctype>
3  using namespace std;
4  int main() {
5      char c;
6      cin >> c;
7      if(isdigit(c)) {
8          cout << "yes";
9      } else {
10         cout << "no";
11     }
12     return 0;
13 }
```

## ALGO-69. 字符串逆序

输入一个字符串，长度在100以内，按相反次序输出其中的所有字符。

样例输入

tsinghua

样例输出

auhgnist

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s;
5      cin >> s;
6      for(int i = s.length() - 1; i >= 0; i--) {
7          cout << s[i];
8      }
9      return 0;
10 }
```

## ALGO-70. 最长字符串

求出5个字符串中最长的字符串。每个字符串长度在100以内，且全为小写字母。

样例输入

one two three four five

样例输出

three

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string a[5];
5      int m = 0;
6      int j = -1;
7      for(int i = 0; i < 5; i++) {
8          cin >> a[i];
9          if(a[i].length() > m) {
10             m = a[i].length();
11             j = i;
12         }
13     }
14     cout << a[j];
15     return 0;
16 }
```



## ALGO-71. 比较字符串

编程实现两个字符串s1和s2的字典序比较。（保证每一个字符串不是另一个的前缀，且长度在100以内）。若s1和s2相等，输出0；若它们不相等，则指出其第一个不同字符的ASCII码的差值：如果s1>s2，则差值为正；如果s1<s2，则差值为负。

样例输入

java basic

样例输出

8

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string s1, s2;
6      cin >> s1 >> s2;
7      for (int i = 0; i < s1.length() && i < s2.length(); i++) {
8          if (s1[i] != s2[i]) {
9              cout << s1[i] - s2[i] << endl;
10             return 0;
11         }
12     }
13     cout << 0 << endl;
14     return 0;
15 }
```

## ALGO-72. 成绩的等级输出

输入一个百分制的成绩t后，按下式输出它的等级。等级为：90~100为A，80~89为B，70~79为C，60~69为D，0~59为E。

样例输入

98

样例输出

A

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      double n;
5      if(n >= 90) {
6          cout << "A";
7      } else if(n >= 80) {
8          cout << "B";
9      } else if(n >= 70) {
```

```

10         cout << "C";
11     } else if(n >= 60) {
12         cout << "D";
13     } else {
14         cout << "E";
15     }
16     return 0;
17 }

```

### ALGO-73. 统计字符次数

输入一个字符串(长度在100以内), 统计其中数字字符出现的次数。

样例输入

Ab100cd200

样例输出

6

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s;
5      cin >> s;
6      int cnt = 0;
7      for(int i = 0; i < s.length(); i++) {
8          if(isdigit(s[i]))
9              cnt++;
10     }
11     cout << cnt;
12     return 0;
13 }

```

### ALGO-74. 连接字符串

编程将两个字符串连接起来。例如country与side相连接成为countryside。

输入两行, 每行一个字符串 (只包含小写字母, 长度不超过100); 输出一行一个字符串。

样例输入

country

side

样例输出

countryside

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s1, s2;
5      cin >> s1 >> s2;
6      cout << s1 << s2;
7      return 0;
8  }

```

## ALGO-75. 筛选号码

### 问题描述

有n个人围成一圈，顺序排号（编号为1到n）。从第1个人开始报数(从1到3报数)，凡报到3的人退出圈子。从下一个人开始继续报数，直到剩下最后一个人，游戏结束。

问最后留下的是原来第几号的那位。

举个例子，8个人围成一圈：

1 2 3 4 5 6 7 8

第1次报数之后，3退出，剩下：

1 2 4 5 6 7 8 （现在从4开始报数）

第2次报数之后，6退出，剩下：

1 2 4 5 7 8 （现在从7开始报数）

第3次报数之后，1退出，剩下：

2 4 5 7 8 （现在从2开始报数）

第4次报数之后，5退出，剩下：

2 4 7 8 （现在从7开始报数）

第5次报数之后，2退出，剩下：

4 7 8 （现在从4开始报数）

第6次报数之后，8退出，剩下：

4 7 （现在从4开始报数）

最后一次报数之后，4退出，剩下：

7.

所以，最后留下来的人编号是7。

### 输入格式

一个正整数n, ( $1 < n < 10000$ )

### 输出格式

一个正整数，最后留下来的那个人的编号。

### 样例输入

8

### 样例输出

7

### 数据规模和约定

对于100%的数据,  $1 < n < 10000$ 。

分析: 使用双向队列, 每次把队头的元素取出, 根据cnt计数判断要不要取出, 不需要取出就扔到队尾  
~

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4  int main() {
5      deque<int> dq;
6      int n, cnt = 1;
7      cin >> n;
8      for (int i = 1; i <= n; i++)
9          dq.push_back(i);
10     while (dq.size() > 1) {
11         int t = dq.front();
12         dq.pop_front();
13         if (cnt == 1 || cnt % 3 != 0)
14             dq.push_back(t);
15         cnt++;
16     }
17     cout << dq.front();
18     return 0;
19 }
```

## ALGO-76. 十进制数转八进制数

编写函数把一个十进制数输出其对应的八进制数。

样例输入

9274

样例输出

22072

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      int n;
6      cin >> n;
7      string s;
8      while(n) {
9          s = (char)(n % 8 + '0') + s;
10         n = n / 8;
11     }
12     cout << s;
13     return 0;
14 }
```

## ALGO-77. 斜率计算

输入两个点的坐标，即 $p_1 = (x_1, y_1)$ 和 $p_2 = (x_2, y_2)$ ，求过这两个点的直线的斜率。如果斜率为无穷大输出“INF”。

样例输入

1 2  
2 4

样例输出

2

样例输入

1 2  
1 4

样例输出

INF

样例输入

1 2  
3 2

样例输出

0

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b, c, d;
5      cin >> a >> b >> c >> d;
6      if (a == c) cout << "INF\n";
7      else cout << (b - d) / (a - c);
8      return 0;
9  }
```

## ALGO-78. 确定元音字母位置

输入一个字符串，编写程序输出该字符串中元音字母的首次出现位置，如果没有元音字母输出0。英语元音字母只有'a'、'e'、'i'、'o'、'u'五个。

样例输入：

hello

样例输出：

2

样例输入:

apple

样例输出:

1

样例输入:

pmp

样例输出:

0

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s;
5      cin >> s;
6      for(int i = 0; i < s.length(); i++) {
7          if(s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' ||
8             s[i] == 'u' || s[i] == 'A' || s[i] == 'E' || s[i] == 'I' || s[i] == 'O'
9             || s[i] == 'U') {
10             cout << i+1;
11             return 0;
12         }
13     }
14     cout << 0;
15     return 0;
16 }
```

## ALGO-79. 删除数组零元素

从键盘读入n个整数放入数组中，编写函数CompactIntegers，删除数组中所有值为0的元素，其后元素向数组首端移动。注意，CompactIntegers函数需要接受数组及其元素个数作为参数，函数返回值应为删除操作执行后数组的新元素个数。输出删除后数组中元素的个数并依次输出数组元素。

样例输入: (输入格式说明: 5为输入数据的个数, 3 4 0 0 2 是以空格隔开的5个整数)

5

3 4 0 0 2

样例输出: (输出格式说明: 3为非零数据的个数, 3 4 2 是以空格隔开的3个非零整数)

3

3 4 2

样例输入:

7

0070090

样例输出:

2

79

样例输入:

3

000

样例输出:

0

```
1  #include <iostream>
2  using namespace std;
3
4  int CompactIntegers(int arr[],int n) {
5      int count = n;
6      for (int i = 0; i < n; i++) {
7          if (arr[i] == 0) {
8              count--;
9          }
10     }
11     return count;
12 }
13
14 int main() {
15     int n;
16     cin >> n;
17     int *a = new int [n];
18     for (int i = 0; i < n; i++) {
19         cin >> a[i];
20         int temp = a[i];
21     }
22     cout << CompactIntegers(a, n) << endl;
23     for (int i = 0; i < n; i++) {
24         if (a[i] != 0)
25             cout << a[i] << " ";
26     }
27     delete [] a;
28     return 0;
29 }
```

## ALGO-80. 整数平均值

编写函数，求包含n个元素的整数数组中元素的平均值。要求在函数内部使用指针操纵数组元素，其中n个整数从键盘输入，输出为其平均值。

样例输入：（输入格式说明：5为输入数据的个数，3 4 0 0 2 是以空格隔开的5个整数）

5

3 4 0 0 2

样例输出：

1

样例输入：

7

3 2 7 5 2 9 1

样例输出：

4

```
1  #include <iostream>
2  char c[1000][1000];
3  using namespace std;
4  int main() {
5      int n, sum = 0, a[1000];
6      cin >> n;
7      for(int i = 0; i < n; i++)
8          cin >> a[i];
9      for(int i = 0; i < n; i++)
10         sum += *(a+i);
11     cout << sum / n;
12     return 0;
13 }
```

## ALGO-81. 动态数组使用

从键盘读入n个整数，使用动态数组存储所读入的整数，并计算它们的和与平均值分别输出。要求尽可能使用函数实现程序代码。平均值为小数的只保留其整数部分。

样例输入：

5

3 4 0 0 2

样例输出：

9 1

样例输入：

7



3 2 7 5 2 9 1

样例输出:

29 4

```
1  #include <iostream>
2  using namespace std;
3  int add(int arr[], int n) {
4      int sum = 0;
5      for (int i = 0; i < n; i++) {
6          sum = sum + arr[i];
7      }
8      return sum;
9  }
10
11 int main() {
12     int n;
13     cin >> n;
14     int *a = new int [n];
15     for (int i = 0; i < n; i++) {
16         cin >> a[i];
17     }
18     cout << add(a, n) << " ";
19     int aver = 0;
20     aver = add(a, n) / n;
21     cout << aver;
22     delete [] a;
23     return 0;
24 }
```

## ALGO-82. 输出米字形

根据输入的正整数 $n$  ( $1 \leq n \leq 26$ ) 米字形由一个 $(2n-1) \times (2n-1)$ 的矩阵组成，矩阵包含从大写A开始的 $n$ 个字母  
例如: $n=3$ 时，包含A,B,C;  $n=4$ 时，包含A,B,C,D。

矩阵的正中间为 $n$ 个字母中字典序最大的那个，从这个字母开始，沿着西北、正北、东北、正西、正东、西南、正南、东南八个方向各有一条由大写字母组成的直线。并且直线上的字母按字典序依次减小，直到大写字母A。

矩阵的其它位置用英文句号。填充。

样例输入一

3

样例输出一

A. A. A

. B B B .

A B C B A

. B B B.

A. A. A

样例输入二

4

样例输出二

A. . A. . A

. B. B. B.

. . C C C. .

A B C D C B A

. . C C C. .

. B. B. B.

A. . A. . A

```
1  #include <iostream>
2  #include <algorithm>
3  char c[1000][1000];
4  using namespace std;
5  int main() {
6      fill(c[0],c[0]+1000*1000, '.');
7      int n;
8      cin >> n;
9      for(int i = 0; i < n; i++){
10         char ch = 'A' + n - i - 1;
11         c[n+i][n+i] = c[n-i][n-i] = c[n+i][n-i] = c[n-i][n+i] = ch;
12         c[n][n+i] = c[n][n-i] = c[n+i][n] = c[n-i][n] = ch;
13     }
14     for(int i = 1; i <= 2*n-1; i++){
15         for(int j = 1; j <= 2*n-1; j++){
16             cout << c[i][j];
17             cout << endl;
18         }
19     }
20     return 0;
}
```

## ALGO-83. 阶乘

### 问题描述

一个整数 $n$ 的阶乘可以写成 $n!$ ，它表示从1到 $n$ 这 $n$ 个整数的乘积。阶乘的增长速度非常快，例如， $13!$ 就已经比较大了，已经无法存放在一个整型变量中；而 $35!$ 就更大了，它已经无法存放在一个浮点型变量中。因此，当 $n$ 比较大时，去计算 $n!$ 是非常困难的。幸运的是，在本题中，我们的任务不是去计算 $n!$ ，而是去计算 $n!$ 最右边的那个非0的数字是多少。例如， $5! = 1*2*3*4*5 = 120$ ，因此 $5!$ 最右边的那个非0的数字是2。再如： $7! = 5040$ ，因此 $7!$ 最右边的那个非0的数字是4。请编写一个程序，输入一个

整数 $n(n \leq 100)$ ，然后输出 $n!$  最右边的那个非0的数字是多少。

输入格式：输入只有一个整数 $n$ 。

输出格式：输出只有一个整数，即 $n!$  最右边的那个非0的数字。

输入输出样例

样例输入

6

样例输出

2

分析：1.取出每个数字因子2和5的个数，把剩余的数字乘积去最后以为循环此操作

2.最后根据2和5因子的个数判断非0的结尾～

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int n, cnt2 = 0, cnt5 = 0, ans = 1;
6      cin >> n;
7      for(int i = 1; i <= n; i++){
8          int t = i;
9          while(t % 2 == 0){
10             cnt2++;
11             t /= 2;
12         }
13         while(t % 5 == 0){
14             cnt5++;
15             t /= 5;
16         }
17         ans = ans * t % 10;
18     }
19     if(cnt2 >= cnt5) ans *= pow(2, cnt2 - cnt5);
20     else ans *= pow(5, cnt5 - cnt2);
21     cout << ans % 10 << endl;
22     return 0;
23 }
```

## ALGO-84. 大小写转换

问题描述

编写一个程序，输入一个字符串（长度不超过20），然后把这个字符串内的每一个字符进行大小写变换，即将大写字母变成小写，小写字母变成大写，然后把这个新的字符串输出。

输入格式：输入一个字符串，而且这个字符串当中只包含英文字母，不包含其他类型的字符，也没有空格。

输出格式：输出经过转换后的字符串。

样例输入

AeDb

样例输出

aEdB

```
1  #include <iostream>
2  #include <cctype>
3  #include <string>
4  using namespace std;
5  int main() {
6      string s;
7      cin >> s;
8      int len = s.length();
9      for (int i = 0; i < len; i++) {
10         if (s[i] >= 'A' && s[i] <= 'Z')
11             s[i] = tolower(s[i]);
12         else
13             s[i] = toupper(s[i]);
14     }
15     cout << s;
16 }
```

## ALGO-85. 进制转换

问题描述

编写一个程序，输入一个二进制的字符串（长度不超过32），然后计算出相应的十进制整数，并把它打印出来。

输入格式：输入为一个字符串，每个字符都是'0'或'1'，字符串的长度不超过32。

输出格式：输出一个整数。

输入输出样例

样例输入

1101

样例输出

13

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      string s;
6      cin >> s;
7      int len = s.length() - 1;
```

```

8     int cnt = 0;
9     for(int i = len; i >= 0; i--) {
10         if(s[i] == '1') {
11             cnt += pow(2, len - i);
12         }
13     }
14     cout << cnt;
15     return 0;
16 }

```

## ALGO-86. 矩阵乘法

### 问题描述

输入两个矩阵，分别是 $m*s$ ， $s*n$ 大小。输出两个矩阵相乘的结果。

### 输入格式

第一行，空格隔开的三个正整数 $m,s,n$ （均不超过200）。  
 接下来 $m$ 行，每行 $s$ 个空格隔开的整数，表示矩阵A（ $i, j$ ）。  
 接下来 $s$ 行，每行 $n$ 个空格隔开的整数，表示矩阵B（ $i, j$ ）。

### 输出格式

$m$ 行，每行 $n$ 个空格隔开的整数，输出相乘後的矩阵C（ $i, j$ ）的值。

### 样例输入

```

2 3 2
1 0 -1
1 1 -3
0 3
1 2
3 1

```

### 样例输出

```

-3 2
-8 2

```

```

1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  vector<int> f(vector<int> v, int k, int m) {
7      vector<int> ans(5);
8      if (k == 0) {
9          ans[1] = ans[4] = 1;
10         ans[2] = ans[3] = 0;
11     } else {
12         if (k % 2 == 0) {

```

```

13         vector<int> t = f(v, k / 2, m);
14         ans[1] = (t[1] * t[1] + t[2] * t[3]) % m;
15         ans[2] = (t[1] * t[2] + t[2] * t[4]) % m;
16         ans[3] = (t[3] * t[1] + t[4] * t[3]) % m;
17         ans[4] = (t[3] * t[2] + t[4] * t[4]) % m;
18     } else {
19         vector<int> t = f(v, k - 1, m);
20         ans[1] = (t[1] * v[1] + t[2] * v[3]) % m;
21         ans[2] = (t[1] * v[2] + t[2] * v[4]) % m;
22         ans[3] = (t[3] * v[1] + t[4] * v[3]) % m;
23         ans[4] = (t[3] * v[2] + t[4] * v[4]) % m;
24     }
25 }
26 return ans;
27 }
28
29 int main() {
30     int k, m, a, b, c, d;
31     vector<int> v(5), ans;
32     cin >> k >> m >> v[1] >> v[2] >> v[3] >> v[4];
33     ans = f(v, k, m);
34     printf("%d %d\n%d %d\n", ans[1], ans[2], ans[3], ans[4]);
35     return 0;
36 }

```

## ALGO-86. 矩阵乘法

### 问题描述

输入两个矩阵，分别是 $m \times s$ ， $s \times n$ 大小。输出两个矩阵相乘的结果。

### 输入格式

第一行，空格隔开的三个正整数 $m, s, n$ （均不超过200）。

接下来 $m$ 行，每行 $s$ 个空格隔开的整数，表示矩阵 $A$ （ $i, j$ ）。

接下来 $s$ 行，每行 $n$ 个空格隔开的整数，表示矩阵 $B$ （ $i, j$ ）。

### 输出格式

$m$ 行，每行 $n$ 个空格隔开的整数，输出相乘後的矩阵 $C$ （ $i, j$ ）的值。

### 样例输入

2 3 2

1 0 -1

1 1 -3

0 3

1 2

样例输出

-3 2

-8 2

提示

矩阵C应该是m行n列，其中C(i,j)等于矩阵A第i行行向量与矩阵B第j列列向量的内积。

例如样例中 $C(1,1)=(1,0,-1)*(0,1,3) = 1 * 0 + 0*1+(-1)*3=-3$

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int main() {
5      int m, s, n;
6      cin >> m >> s >> n;
7      vector< vector<int> > a(m, vector<int>(s));
8      vector< vector<int> > b(s, vector<int>(n));
9      vector< vector<int> > c(m, vector<int>(n));
10     for (int i = 0; i < m; i++) {
11         for (int j = 0; j < s; j++) {
12             cin >> a[i][j];
13         }
14     }
15     for (int i = 0; i < s; i++) {
16         for (int j = 0; j < n; j++) {
17             cin >> b[i][j];
18         }
19     }
20     for (int i = 0; i < m; i++) {
21         for (int j = 0; j < n; j++) {
22             for (int k = 0; k < s; k++) {
23                 c[i][j] = c[i][j] + a[i][k] * b[k][j];
24             }
25         }
26     }
27     for (int i = 0; i < m; i++) {
28         for (int j = 0; j < n; j++){
29             cout << c[i][j] << " ";
30         }
31         cout << endl;
32     }
33     return 0;
34 }
```

## ALGO-87. 字串统计

## 问题描述

给定一个长度为 $n$ 的字符串 $S$ ，还有一个数字 $L$ ，统计长度大于等于 $L$ 的出现次数最多的子串（不同的出现可以相交），如果有多个，输出最长的，如果仍然有多个，输出第一次出现最早的。

## 输入格式

第一行一个数字 $L$ 。

第二行是字符串 $S$ 。

$L$ 大于0，且不超过 $S$ 的长度。

## 输出格式

一行，题目要求的字符串。

## 输入样例1：

4

bbaabbaaaaa

## 输出样例1：

bbaa

## 输入样例2：

2

bbaabbaaaaa

## 输出样例2：

aa

## 数据规模和约定

$n \leq 60$

$S$ 中所有字符都是小写英文字母。

## 提示

枚举所有可能的子串，统计出现次数，找出符合条件的那个

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4  using namespace std;
5  int main() {
6      int num;
7      cin >> num;
8      string a;
9      cin >> a;
10     vector<string> v;
11     int len = a.length();
```



```

12     for (int i = num; i <= len; i++) {
13         for (int j = 0; j <= len - i; j++) {
14             v.push_back(a.substr(j, i));
15         }
16     }
17
18     int max = 0;
19     int k = 0;
20     vector<int> book(v.size());
21     for (int i = 0; i <= v.size() - 1; i++) {
22         for (int j = 0; j <= v.size() - 1; j++) {
23             if (i != j && v[i] == v[j])
24                 book[i]++;
25         }
26     }
27
28     for (int i = 0; i <= v.size() - 1; i++) {
29         if (book[i] > max || (book[i] == max && v[k].length() <
30             v[i].length())) {
31             k = i;
32             max = book[i];
33         }
34     }
35     cout << v[k];
36     return 0;
37 }

```

## ALGO-88. 字串统计

### 问题描述

给定一个长度为n的字符串S，还有一个数字L，统计长度大于等于L的出现次数最多的子串（不同的出现可以相交），如果有多个，输出最长的，如果仍然有多个，输出第一次出现最早的。

### 输入格式

第一行一个数字L。

第二行是字符串S。

L大于0，且不超过S的长度。

### 输出格式

一行，题目要求的字符串。

### 输入样例1：

4

bbaabbaaaaa

### 输出样例1：

bbaa

输入样例2:

2

bbaabbabaaaa

输出样例2:

aa

数据规模和约定

$n \leq 60$

S中所有字符都是小写英文字母。

提示

枚举所有可能的子串，统计出现次数，找出符合条件的那个

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4  using namespace std;
5  int main() {
6      int num;
7      cin >> num;
8      string a;
9      cin >> a;
10     vector<string> v;
11     int len = a.length();
12     for (int i = num; i <= len; i++) {
13         for (int j = 0; j <= len - i; j++) {
14             v.push_back(a.substr(j, i));
15         }
16     }
17
18     int max = 0;
19     int k = 0;
20     vector<int> book(v.size());
21     for (int i = 0; i <= v.size() - 1; i++) {
22         for (int j = 0; j <= v.size() - 1; j++) {
23             if (i != j && v[i] == v[j])
24                 book[i]++;
25         }
26     }
27
28     for (int i = 0; i <= v.size() - 1; i++) {
29         if (book[i] > max || (book[i] == max && v[k].length() <
30             v[i].length())) {
31             k = i;
32         }
33     }
```

```

31         max = book[i];
32     }
33 }
34 cout << v[k];
35 return 0;
36 }

```

## ALGO-89. 字符删除

### 问题描述

编写一个程序，先输入一个字符串str（长度不超过20），再输入单独的一个字符ch，然后程序会把字符串str当中出现的所有的ch字符都删掉，从而得到一个新的字符串str2，然后把这个字符串打印出来。

输入格式：输入有两行，第一行是一个字符串（内部没有空格），第二行是一个字符。

输出格式：经过处理以后的字符串。

输入输出样例

样例输入

123-45-678

-

样例输出

12345678

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s;
5      char c;
6      getline(cin, s);
7      cin >> c;
8      for (int i = 0; i < s.size(); i++)
9          if (s[i] != c) cout << s[i];
10     cout << endl;
11     return 0;
12 }

```

## ALGO-90. 出现次数最多的整数

### 问题描述

编写一个程序，读入一组整数，这组整数是按照从小到大的顺序排列的，它们的个数N也是由用户输入的，最多不会超过20。

然后程序将对这个数组进行统计，把出现次数最多的那个数组元素值打印出来。如果有两个元素值出现的次数相同，即并列第一，

那么只打印比较小的那个值。

输入格式：第一行是一个整数N， $N \leq 20$ ；接下来有N行，每一行表示一个整数，并且按照从小到大的顺序排列。

输出格式：输出只有一行，即出现次数最多的那个元素值。

输入输出样例

样例输入

5  
100  
150  
150  
200  
250

样例输出

150

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      if (n > 0) {
7          int *a = new int [n];
8          int b, count = 1, temp = 1;
9          cin >> a[0];
10         b = a[0];
11         for(int i = 1; i < n; i++) {
12             cin >> a[i];
13             if (a[i] == a[i - 1]) {
14                 temp++;
15                 if (temp > count) {
16                     count = temp;
17                     b = a[i];
18                 }
19             } else {
20                 temp = 1;
21             }
22         }
23         cout << b;
24         delete [] a;
25     }
26     return 0;
27 }
```

## ALGO-91. Anagrams问题

### 问题描述

Anagrams指的是具有如下特性的两个单词：在这两个单词当中，每一个英文字母（不区分大小写）所出现的次数都是相同的。例如，“Unclear”和“Nuclear”、“Rimon”和“MinOR”都是Anagrams。编写一个程序，输入两个单词，然后判断一下，这两个单词是否是Anagrams。每一个单词的长度不会超过80个字符，而且是大小写无关的。

输入格式：输入有两行，分别为两个单词。

输出格式：输出只有一个字母Y或N，分别表示Yes和No。

输入输出样例

样例输入

Unclear

Nuclear

样例输出

Y

```
1  #include <iostream>
2  #include <string>
3  #include <cctype>
4  using namespace std;
5  int main() {
6      int a[26] = {0};
7      int b[26] = {0};
8      int flag = 1;
9      string m, n;
10     cin >> m;
11     cin >> n;
12     int lenm = m.length();
13     int lenn = n.length();
14     if (lenm != lenn) {
15         cout << 'N';
16         return 0;
17     }
18     for (int i = 0; i < lenm; i++) {
19         m[i] = toupper(m[i]);
20         n[i] = toupper(n[i]);
21     }
22     for (int i = 0; i < lenm; i++) {
23         a[m[i] - 'A']++;
24         b[n[i] - 'A']++;
25     }
26     for (int i = 0; i < 26; i++) {
27         if (a[i] != b[i])
28             flag = 0;
```

```

29     }
30     if (flag == 0)
31         cout << 'N';
32     else
33         cout << 'Y';
34     return 0;
35 }

```

## ALGO-92. 前缀表达式

### 问题描述

编写一个程序，以字符串方式输入一个前缀表达式，然后计算它的值。

输入格式为：“运算符 对象1 对象2”，其中，运算符为“+”（加法）、“-”（减法）、“\*”（乘法）或“/”（除法），运算对象为不超过10的整数，它们之间用一个空格隔开。

要求：对于加、减、乘、除这四种运算，分别设计相应的函数来实现。

输入格式：输入只有一行，即一个前缀表达式字符串。

输出格式：输出相应的计算结果（如果是除法，直接采用c语言的“/”运算符，结果为整数）。

输入输出样例

### 样例输入

+ 5 2

### 样例输出

7

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  void add(int a, int b) {cout << a + b;}
5  void min(int a, int b) {cout << a - b;}
6  void mul(int a, int b) {cout << a * b;}
7  void div1(int a, int b) {cout << a / b;}
8
9  int main() {
10     string s;
11     int a, b;
12     getline(cin, s);
13     if (s[3] == ' ') {
14         a = s[2] - '0';
15         if (s[5] == '\\0') {
16             b = s[4] - '0';
17         }
18     }
19     else {
20         b = 10;

```

```

20     }
21 }
22 else {
23     a = 10;
24     if (s[6] == '\0') {
25         b = s[5] - '0';
26     }
27     else {
28         b = 10;
29     }
30 }
31 switch(s[0]) {
32     case '+':add(a, b);break;
33     case '-':min(a, b);break;
34     case '*':mul(a, b);break;
35     case '/':div1(a, b);break;
36 }
37 return 0;
38 }

```

## ALGO-93. 反置数

### 问题描述

一个整数的“反置数”指的是把该整数的每一位数字的顺序颠倒过来所得到的另一个整数。如果一个整数的末尾是以0结尾，那么在它的反置数当中，这些0就被省略掉了。比如说，1245的反置数是5421，而1200的反置数是21。请编写一个程序，输入两个整数，然后计算这两个整数的反置数之和sum，然后再把sum的反置数打印出来。要求：由于在本题中需要多次去计算一个整数的反置数，因此必须把这部分代码抽象为一个函数的形式。

输入格式：输入只有一行，包括两个整数，中间用空格隔开。

输出格式：输出只有一行，即相应的结果。

输入输出样例

样例输入

435 754

样例输出

199

```

1  #include <iostream>
2  using namespace std;
3  int re(int n){
4      int ans = 0;
5      while(n){
6          ans = ans * 10 + n % 10;
7          n /= 10;
8      }
9      return ans;
10 }

```

```

11  int main() {
12      int a, b;
13      cin >> a >> b;
14      cout << re(re(a)+re(b));
15      return 0;
16  }

```

## ALGO-94. 新生舞会

### 问题描述

新生舞会开始了。n名新生每人有三个属性：姓名、学号、性别。其中，姓名用长度不超过20的仅由大小写字母构成的字符串表示，学号用长度不超过10的仅由数字构成的字符串表示，性别用一个大写字母'F'或'M'表示。任意两人的姓名、学号均互不相同。换言之，每个人可被其姓名或学号唯一确定。给出m对两人的信息（姓名或学号），判断他们是否能共舞。两人能共舞的充要条件为两人性别相异。

### 输入

第一行一个整数n ( $2 \leq n \leq 1000$ )，表示学生人数。接下来的n行每行依次包含一名新生的姓名、学号、性别，分别用一个空格隔开。

之后的一行是一个整数m ( $1 \leq m \leq 1000$ )，表示询问的数目。接着的m行每行包含两个信息（姓名或学号），保证两个信息不属于同一人，中间用一个空格隔开。

### 输出

对于每个询问输出一行，如果两人可以共舞，输出一个大写字母'Y'，否则输出一个大写字母'N'。

### 样例输入

```

4
John 10 M
Jack 11 M
Kate 20 F
Jim 21 M
3
John 11
20 Jack
Jim Jack

```

### 样例输出

```

N
Y
N

```

分析：1.用map存储名字对应学号，学号对应名字，以及名字和学号分别对应性别  
2.判断是不是本人和性别是否为异性即可～

```

1  #include <iostream>
2  #include <map>

```



```

3  using namespace std;
4  int main() {
5      map<string, string> sex;
6      map<string, string> id_name;
7      int n, k;
8      cin >> n;
9      for(int i = 0; i < n; i++){
10         string id, name, tsex;
11         cin >> id >> name >> tsex;
12         id_name[id] = name;
13         id_name[name] = id;
14         sex[id] = sex[name] = tsex;
15     }
16     cin >> k;
17     for(int i = 0; i < k; i++){
18         string s1, s2;
19         cin >> s1 >> s2;
20         if(id_name[s1] != s2 && sex[s1] != sex[s2]) cout << "Y\n";
21         else cout << "N\n";
22     }
23     return 0;
24 }

```

## ALGO-95. 2的次幂表示

### 问题描述

任何一个正整数都可以用2进制表示，例如：137的2进制表示为10001001。

将这种2进制表示写成2的次幂的和的形式，令次幂高的排在前面，可得到如下表达式：

$$137=2^7+2^3+2^0$$

现在约定幂次用括号来表示，即 $a^b$ 表示为a (b)

此时，137可表示为：2 (7) +2 (3) +2 (0)

进一步：7=2<sup>2</sup>+2+2<sup>0</sup> (2<sup>1</sup>用2表示)

$$3=2+2^0$$

所以最后137可表示为：2 (2 (2) +2+2 (0) ) +2 (2+2 (0) ) +2 (0)

$$\text{又如：} 1315=2^{10}+2^8+2^5+2+1$$

所以1315最后可表示为：

$$2 (2 (2+2 (0) ) +2) +2 (2 (2+2 (0) ) ) +2 (2 (2) +2 (0) ) +2+2 (0)$$

### 输入格式

正整数 (1≤n≤20000)

### 输出格式

符合约定的n的0, 2表示 (在表示中不能有空格)

样例输入

137

样例输出

2(2(2)+2+2(0))+2(2+2(0))+2(0)

样例输入

1315

样例输出

2(2(2+2(0))+2)+2(2(2+2(0)))+2(2(2)+2(0))+2+2(0)

提示

用递归实现会比较简单，可以一边递归一边输出

```
1  #include <iostream>
2  using namespace std;
3
4  void func(int a);
5
6  int main() {
7      int n;
8      cin >> n;
9      func(n);
10     return 0;
11 }
12
13 void func(int a) {
14     int i = 0;
15     int s[100] = {0};
16     if(a == 0) {
17         cout << "";
18     }
19     else {
20         while (a != 0) {
21             s[i++] = a % 2;
22             a = a / 2;
23         }
24         int t = 0;
25         for (int c = 0; c <= i - 1; c++) {
26             if (s[c] == 1) {
27                 t = c; break;
28             }
29         }
30         for (int j = i - 1; j >= t + 1; j--) {
31             if (s[j] != 0) {
32                 if (j == 1) {
33                     cout << "2+";
```

```

34         }
35         else if (j == 2) {
36             cout << "2(2)+";
37         }
38         else {
39             cout << "2(";
40             func (j);
41             cout << ")+";
42         }
43     }
44 }
45 if (t == 0) {
46     cout << "2(0)";
47 }
48 else if (t == 1) {
49     cout << "2";
50 }
51 else if (t == 2) {
52     cout << "2(2)";
53 }
54 else {
55     cout << "2(";
56     func(t);
57     cout << ")";
58 }
59 }
60 }
61 }

```

## ALGO-96. Hello World!

### 描述

本题定义本学期作业题的输出格式，请认真阅读。

如无特殊说明，开头无空格，间隔符为1个空格，答案最后必须输出换行符("\n")。

### 输入格式

无

### 输出格式

Hello World!

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      cout << "Hello World!" << endl;
5      return 0;
6  }

```

## ALGO-97. 排序

### 问题描述

编写一个程序，输入3个整数，然后程序将对这三个整数按照从大到小进行排列。

输入格式：输入只有一行，即三个整数，中间用空格隔开。

输出格式：输出只有一行，即排序后的结果。

输入输出样例

样例输入

9 2 30

样例输出

30 9 2

### 问题描述

编写一个程序，输入3个整数，然后程序将对这三个整数按照从大到小进行排列。

输入格式：输入只有一行，即三个整数，中间用空格隔开。

输出格式：输出只有一行，即排序后的结果。

输入输出样例

样例输入

9 2 30

样例输出

30 9 2

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b, c;
5      cin >> a >> b >> c;
6      int t;
7      if (a < b) {
8          swap(a, b);
9      }
10     if (a < c) {
11         swap(a, c);
12     }
13     if (b < c) {
14         swap(b, c);
15     }
16     cout << a << " " << b << " " << c;
17     return 0;
18 }
```

## ALGO-98. 数位分离

### 问题描述

编写一个程序，输入一个1000 以内的正整数，然后把这个整数的每一位数字都分离出来，并逐一地显示。

输入格式：输入只有一行，即一个1000以内的正整数。

输出格式：输出只有一行，即该整数的每一位数字，之间用空格隔开。

输入输出样例

样例输入

769

样例输出

7 6 9

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s;
5      cin >> s;
6      for(int i = 0; i < s.length(); i++) {
7          cout << s[i] << " ";
8      }
9      return 0;
10 }
```

## ALGO-99. 薪水计算

### 问题描述

编写一个程序，计算员工的周薪。薪水的计算是以小时为单位，如果在一周的时间内，员工工作的时间不超过40 个小时，那么他/她的总收入等于工作时间乘以每小时的薪水。如果员工工作的时间在40 到50 个小时之间，那么对于前40 个小时，仍按常规方法计算；而对于剩余的超额部分，每小时的薪水按1.5 倍计算。如果员工工作的时间超过了50 个小时，那么对于前40 个小时，仍按常规方法计算；对于40~50 个小时之间的部分，每小时的薪水按1.5 倍计算；而对于超出50 个小时的部分，每小时的薪水按2 倍计算。请编写一个程序，输入员工的工作时间和每小时的薪水，然后计算并显示他/她应该得到的周薪。

输入格式：输入只有一行，包括一个整数和一个实数，分别表示工作时间和每小时薪水。

输出格式：输出只有一个实数，表示周薪，保留小数点后2位。

输入输出样例

样例输入

40 50

样例输出

2000.00

```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int n;
6      double sal, sum;
7      cin >> n >> sal;
8      sum = min(40, n) * sal + min(10, max(0, n - 40)) * sal * 1.5 +
max(0, n - 50) * sal * 2;
9      printf("%.2f", sum);
10     return 0;
11 }

```

## ALGO-100. 整除问题

### 问题描述

编写一个程序，输入三个正整数min、max和factor，然后对于min到max之间的每一个整数（包括min和max），如果它能被factor整除，就把它打印出来。

输入格式：输入只有一行，包括三个整数min、max和factor。

输出格式：输出只有一行，包括若干个整数。

输入输出样例

样例输入

1 10 3

样例输出

3 6 9

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int l, r, n;
5      cin >> l >> r >> n;
6      for(int i = l; i <= r; i++)
7          if(i % n == 0) cout << i << " ";
8      return 0;
9  }

```

## ALGO-101. 图形显示

### 问题描述

编写一个程序，首先输入一个整数，例如5，然后在屏幕上显示如下的图形（5表示行数）：

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      for (int i = 0; i < n; i++){
7          for (int j = 0; j < n - i - 1; j++) {
8              cout << "*" << " ";
9          }
10         cout << "*" << endl;
11     }
12     return 0;
13 }
```

## ALGO-102. 数对

### 问题描述

编写一个程序，该程序从用户读入一个整数，然后列出所有的数对，每个数对的乘积即为该数。

输入格式：输入只有一行，即一个整数。

输出格式：输出有若干行，每一行是一个乘法式子。（注意：运算符与数字之间有一个空格）

### 样例输入

32

### 样例输出

1 \* 32 = 32

2 \* 16 = 32

4 \* 8 = 32

8 \* 4 = 32

16 \* 2 = 32

32 \* 1 = 32

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      for(int i = 1; i <= n; i++){
7          if(n % i == 0){
8              printf("%d * %d = %d\n", i, n / i, n);
9          }
10     }
11     return 0;
12 }

```

## ALGO-103. 完数

### 问题描述

一个数如果恰好等于它的因子之和，这个数就称为“完数”。例如，6的因子为1、2、3，而 $6=1+2+3$ ，因此6就是“完数”。又如，28的因子为1、2、4、7、14，而 $28=1+2+4+7+14$ ，因此28也是“完数”。编写一个程序，判断用户输入的一个数是否为“完数”。

输入格式：输入只有一行，即一个整数。

输出格式：输出只有一行，如果该数为完数，输出yes，否则输出no。

输入输出样例

样例输入

6

样例输出

yes

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int sum = 0;
7      for(int i = 1; i <= n/2; i++) {
8          if(n % i == 0)
9              sum += i;
10     }
11     if(sum == n)
12         cout << "yes";
13     else
14         cout << "no";
15     return 0;
16 }

```



## ALGO-104. 阿尔法乘积

### 问题描述

计算一个整数的阿尔法乘积。对于一个整数x来说，它的阿尔法乘积是这样来计算的：如果x是一个个位数，那么它的阿尔法乘积就是它本身；否则的话，x的阿尔法乘积就等于它的各位非0的数字相乘所得到的那个整数的阿尔法乘积。例如：4018224312的阿尔法乘积等于8，它是按照以下的步骤来计算的：

$4018224312 \rightarrow 4*1*8*2*2*4*3*1*2 \rightarrow 3072 \rightarrow 3*7*2 \rightarrow 42 \rightarrow 4*2 \rightarrow 8$

编写一个程序，输入一个正整数（该整数不会超过6,000,000），输出它的阿尔法乘积。

输入格式：输入只有一行，即一个正整数。

输出格式：输出相应的阿尔法乘积。

输入输出样例

样例输入

4018224312

样例输出

8

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s;
5      cin >> s;
6      while(s.length() > 1) {
7          int ans = 1;
8          for(int i = 0; i < s.length(); i++) {
9              if(s[i] != '0') {
10                 ans *= (int)(s[i] - '0');
11             }
12         }
13         s = "";
14         while(ans != 0) {
15             s += (char)(ans % 10 + '0');
16             ans = ans / 10;
17         }
18     }
19     cout << s;
20     return 0;
21 }
```

## ALGO-105. 黑色星期五

### 问题描述

有些西方人比较迷信，如果某个月的13号正好是星期五，他们就会觉得不太吉利，用古人的说法，就是“诸事不宜”。请你编写一个程序，统计出在某个特定的年份中，出现了多少次既是13号又是星期五的情形，以帮助你的迷信朋友解决难题。

说明：（1）一年有365天，闰年有366天，所谓闰年，即能被4整除且不能被100整除的年份，或是既能被100整除也能被400整除的年份；（2）已知1998年1月1日是星期四，用户输入的年份肯定大于或等于1998年。

输入格式：输入只有一行，即某个特定的年份（大于或等于1998年）。

输出格式：输出只有一行，即在这一年中，出现了多少次既是13号又是星期五的情形。

输入输出样例

样例输入

1998

样例输出

3

分析：1.根据基姆拉尔森计算公式枚举即可

2. $W = (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) \% 7$

3.注意：在公式中有个与其他公式不同的地方：把一月和二月看成是上一年的十三月和十四月例：如果是2018-1-1则换算成：2017-13-1来代入公式计算~

```
1  #include <iostream>
2  using namespace std;
3  int ans = 0;
4  void f(int y, int m, int d) {
5      if (m == 1 || m == 2) {
6          y--;
7          m += 12;
8      }
9      if ((d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) %
10 7 == 4 && d == 13) ans++;
11 }
12 int main() {
13     int n;
14     int m31[] = {1, 3, 5, 7, 8, 10, 12};
15     int m30[] = {4, 6, 9, 11};
16     int m2;
17     cin >> n;
18     if (n % 4 == 0 && n % 100 != 0 || n % 400 == 0) {
19         m2 = 29;
20     } else {
21         m2 = 28;
22     }
23     for (int i = 0; i < 7; i++)
24         for (int j = 1; j <= 31; j++)
25             f(n, m31[i], j);
26     for (int i = 0; i < 4; i++)
```

```

27         for (int j = 1; j <= 30; j++)
28             f(n, m30[i], j);
29
30     for (int j = 1; j <= m2; j++)
31         f(n, 2, j);
32     cout << ans << endl;
33     return 0;
34 }

```

## ALGO-106. 6-3判定字符位置

返回给定字符串s中元音字母的首次出现位置。英语元音字母只有'a'、'e'、'i'、'o'、'u'五个。

若字符串中没有元音字母，则返回0。

只考虑小写的情况。

样例输入

and

样例输出

1

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string s, so = "aoieuo";
6      cin >> s;
7      for(int i = 0; i < so.length(); i++){
8          if(so.find(s[i]) != string::npos){
9              cout << i + 1 << endl;
10             return 0;
11         }
12     }
13     cout << 0 << endl;
14     return 0;
15 }

```

## ALGO-107. 9-7链表数据求和操作

读入10个复数，建立对应链表，然后求所有复数的和。

样例输入

1 2

1 3

4 5

2 3

3 1

2 1

4 2

2 2

3 3

1 1

样例输出

23+23i

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int suma = 0, sumb = 0;
5      for(int i = 0; i < 10; i++) {
6          int a, b;
7          cin >> a >> b;
8          suma += a;
9          sumb += b;
10     }
11     cout << suma;
12     if(sumb >= 0) {
13         cout << "+";
14     }
15     cout << sumb << "i";
16     return 0;
17 }
```

## ALGO-110. 字符串的展开

在初赛普及组的“阅读程序写结果”的问题中，我们曾给出一个字符串展开的例子：如果在输入的字符串中，含有类似于“d-h”或者“4-8”的字串，我们就把它当作一种简写，输出时，用连续递增的字母获数字串替代其中的减号，即，将上面两个子串分别输出为“defgh”和“45678”。在本题中，我们通过增加一些参数的设置，使字符串的展开更为灵活。具体约定如下：

(1) 遇到下面的情况需要做字符串的展开：在输入的字符串中，出现了减号“-”，减号两侧同为小写字母或同为数字，且按照ASCII码的顺序，减号右边的字符严格大于左边的字符。

(2) 参数p1：展开方式。p1=1时，对于字母子串，填充小写字母；p1=2时，对于字母子串，填充大写字母。这两种情况下数字子串的填充方式相同。p1=3时，不论是字母子串还是数字子串，都用与要填充的字母个数相同的星号“\*”来填充。

(3) 参数p2：填充字符的重复个数。p2=k表示同一个字符要连续填充k个。例如，当p2=3时，子串“d-h”应扩展为“deeefffgggh”。减号两边的字符不变。

(4) 参数p3：是否改为逆序：p3=1表示维持原来顺序，p3=2表示采用逆序输出，注意这时候仍然不包括减号两端的字符。例如当p1=1、p2=2、p3=2时，子串“d-h”应扩展为“dggffeeh”。

(5) 如果减号右边的字符恰好是左边字符的后继，只删除中间的减号，例如：“d-e”应输出为“de”，“3-4”应输出为“34”。如果减号右边的字符按照ASCII码的顺序小于或等于左边字符，输出时，要保留中间

的减号，例如：“d-d”应输出为“d-d”，“3-1”应输出为“3-1”。

输入格式

输入包括两行：

第1行为用空格隔开的3个正整数，一次表示参数p1，p2，p3。

第2行为一行字符串，仅由数字、小写字母和减号“-”组成。行首和行末均无空格。

输出格式

输出只有一行，为展开后的字符串。

输入输出样例1

输入

1 2 1

abcs-w1234-9s-4zz

输出

abcsttuuvvw1234556677889s-4zz

输入输出样例2

输入

2 3 2

a-d-d

输出

aCCCBBDd-d

输入输出样例3

输入

3 4 2

di-jkstra2-6

输出

dijkstra2\*\*\*\*\*6

数据规模和约定

40%的数据满足：字符串长度不超过5

100%的数据满足：1<=p1<=3，1<=p2<=8，1<=p3<=2。字符串长度不超过100

```
1 #include <iostream>
2 #include <algorithm>
3 #include <string>
4 #include <cctype>
5 using namespace std;
6 int main() {
7     int p1, p2, p3;
8     string s;
9     cin >> p1 >> p2 >> p3 >> s;
10    for (int i = 0; i < s.size(); i++) {
11        cout << s[i];
12        if (i < s.size() - 2 && s[i + 1] == '-' &&
```

```

13         (isdigit(s[i]) && isdigit(s[i + 2]) || islower(s[i]) &&
islower(s[i + 2]))) {
14             char c1 = s[i], c2 = s[i + 2];
15             string ans;
16             if (c1 >= c2) ans = "-";
17             else if (c1 + 1 == c2) ans = "";
18             else {
19                 if (p3 == 1) {
20                     for (int i = c1 + 1; i < c2; i++) {
21                         if (p1 == 1) ans.append(p2, tolower((char) i));
22                         if (p1 == 2) ans.append(p2, toupper((char) i));
23                         if (p1 == 3) ans.append(p2, '*');
24                     }
25                 }
26                 if (p3 == 2) {
27                     for (int i = c2 - 1; i > c1; i--) {
28                         if (p1 == 1) ans.append(p2, tolower((char) i));
29                         if (p1 == 2) ans.append(p2, toupper((char) i));
30                         if (p1 == 3) ans.append(p2, '*');
31                     }
32                 }
33             }
34             cout << ans;
35             i++;
36         }
37     }
38     return 0;
39 }

```

## ALGO-111. 明明的随机数

### 问题描述

明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N个1到1000之间的随机整数（ $N \leq 100$ ），对于其中重复的数字，只保留一个，把其余相同的数去掉，不同的数对应着不同的学生的学号。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作。

### 输入格式

输入有2行，第1行为1个正整数，表示所生成的随机数的个数：

N

第2行有N个用空格隔开的正整数，为所产生的随机数。

### 输出格式

输出也是2行，第1行为1个正整数M，表示不相同的随机数的个数。第2行为M个用空格隔开的正整数，为从小到大排好序的不相同的随机数。

### 样例输入

10

20 40 32 67 40 20 89 300 400 15

样例输出

8

15 20 32 40 67 89 300 400

```
1  #include <iostream>
2  #include <set>
3  using namespace std;
4  int main() {
5      set<int> s;
6      int n;
7      cin >> n;
8      for(int i = 0; i < n; i++) {
9          int temp;
10         cin >> temp;
11         s.insert(temp);
12     }
13     cout << s.size() << endl;
14     for(set<int>::iterator it = s.begin(); it != s.end(); it++) {
15         cout << *it << " ";
16     }
17 }
```

## ALGO-112. 暗恋

问题描述

同在一个高中，他却不敢去找她，虽然在别人看来，那是再简单不过的事。暗恋，是他唯一能做的事。他只能在每天课间操的时候，望望她的位置，看看她倾心的动作，就够了。操场上的彩砖啊，你们的位置，就是他们能够站立的地方，他俩的关系就像砖与砖之间一样固定，无法动摇。还记得当初铺砖的工人，将整个操场按正方形铺砖（整个操场可视为R行C列的矩阵，矩阵的每个元素为一块正方形砖块），正方形砖块有两种，一种为蓝色，另一种为红色。我们定义他和她之间的“爱情指标”为最大纯色正方形的面积，请你写一个程序求出“爱情指标”。

输入格式

第一行两个正整数R和C。

接下来R行C列描述整个操场，红色砖块用1来表示，蓝色砖块用0来表示。

输出格式

一个数，表示他和她之间的“爱情指标”。

样例输入

5 8  
0 0 0 1 1 1 0 1  
1 1 0 1 1 1 1 1  
0 1 1 1 1 1 0 1  
1 0 1 1 1 1 1 0  
1 1 1 0 1 1 0 1

样例输出

9

数据规模和约定

40%的数据R,C≤10;

70%的数据R,C≤50;

100%的数据R,C≤200;

分析：枚举所有方块的最左上角的点，依次向右下角探测，得到能探测到的最大面积，取最大值即可  
~

```
1  #include <iostream>
2  #include <algorithm>
3  #include <cmath>
4  using namespace std;
5  int m[210][210], ans = 0, r, c;
6  int check(int a, int b){
7      for(int k = 1;; k++){
8          for(int i = a; i <= a + k; i++)
9              if(m[i][b+k] != m[a][b]) return k * k;
10         for(int i = b; i <= b + k; i++)
11             if(m[a+k][i] != m[a][b]) return k * k;
12     }
13 }
14 int main() {
15     fill(m[0],m[0]+210*210,-1);
16     scanf("%d%d", &r, &c);
17     for(int i = 1; i <= r; i++)
18         for(int j = 1; j <= c; j++)
19             scanf("%d",&m[i][j]);
20     for(int i = 1; i <= r; i++)
21         for(int j = 1; j <= c; j++)
22             ans = max(ans,check(i,j));
23     cout << ans << endl;
24     return 0;
25 }
```

## ALGO-113. 数的统计

问题描述

在一个有限的正整数序列中，有些数会多次重复出现在这个序列中。



如序列：3，1，2，1，5，1，2。其中1就出现3次，2出现2次，3出现1次，5出现1次。

你的任务是对于给定的正整数序列，从小到大依次输出序列中出现的数及出现的次数。

#### 输入格式

第一行正整数n，表示给定序列中正整数的个数。

第二行是n 个用空格隔开的正整数x，代表给定的序列。

#### 输出格式

若干行，每行两个用一个空格隔开的数，第一个是数列中出现的数，第二个是该数在序列中出现的次数。

#### 样例输入

12

8 2 8 2 2 11 1 1 8 1 13 13

#### 样例输出

1 3

2 3

8 3

11 1

13 2

#### 数据规模和约定

数据：n<=1000；0<x<=1000,000。

```
1  #include <iostream>
2  int a[1000001];
3  using namespace std;
4  int main() {
5      int n;
6      cin >> n;
7      for(int i = 0; i < n; i++) {
8          int temp;
9          cin >> temp;
10         a[temp]++;
11     }
12     for(int i = 0; i <= 1000000; i++) {
13         if(a[i] != 0) {
14             cout << i << " " << a[i] << endl;
15         }
16     }
17     return 0;
18 }
```

## ALGO-114. 黑白无常

### 问题描述

某寝室的同学们在学术完之后准备玩一个游戏：游戏是这样的，每个人头上都被贴了一张白色或者黑色的纸，现在每个人都会说一句话“我看到x张白色纸条和y张黑色的纸条”，又已知每个头上贴着白色纸的人说的是真话、每个头上贴着黑色纸的人说的是谎话，现在要求你判断哪些人头上贴着的是白色的纸条，如果无解输出“NoSolution.”；如果有多组解，则把每个答案中贴白条的人的编号按照大小排列后组成一个数（比如第一个人和第三个人头上贴着的是白纸条，那么这个数就是13；如果第6、7、8个人都贴的是白纸条，那么这个数就是678）输出最小的那个数（如果全部都是黑纸条也满足情况的话，那么输出0）

### 输入格式

第一行为一个整数n，接下来n行中的第i行有两个整数x和y，分别表示第i个人说“我看到x张白色纸条和y张黑色的纸条”。

### 输出格式

一行。如果无解输出“NoSolution.”。否则输出答案中数值（具体见问题描述）最小的那个，如果全部都是黑纸条也满足情况的话，那么输出0

### 样例输入

```
2
1 0
1 0
```

### 样例输出

```
0
```

### 样例输入

```
5
3 1
0 4
1 3
4 0
1 3
```

### 样例输出

```
35
```

### 数据规模和约定

$n \leq 8$

分析：1.枚举所有人的黑白字条情况，判断当前情况是否可能存在，把所有可能情况收集起来输出最小的

2.注意每人看不到自己头上的字条哦～

```
1 #include <iostream>
2 #include <vector>
```

```

3  #include <algorithm>
4  using namespace std;
5  vector<int> ansv;
6  pair<int, int> v[10];
7  int n, k, cnt, a[10];
8  void dfs(int num) {
9      if (num == 0) {
10         int white = 0, black = 0, ans = 0;
11         for (int i = 1; i <= n; i++) {
12             if (a[i] == -1) black++;
13             if (a[i] == 1) white++;
14         }
15         for (int i = 1; i <= n; i++) {
16             if (a[i] == 1 && (v[i].first != white-1 || v[i].second !=
black)) return;
17             if (a[i] == -1 && (v[i].first == white && v[i].second ==
black-1)) return;
18         }
19         for (int i = 1; i <= n; i++)
20             if (a[i] == 1) ans = ans * 10 + i;
21         ansv.push_back(ans);
22     } else {
23         a[num] = -1;
24         dfs(num - 1);
25
26         a[num] = 1;
27         dfs(num - 1);
28     }
29 }
30 int main() {
31     scanf("%d", &n);
32     for (int i = 1; i <= n; i++)
33         scanf("%d%d", &v[i].first, &v[i].second);
34     dfs(n);
35     sort(ansv.begin(), ansv.end());
36     printf("%d\n", ansv[0]);
37     return 0;
38 }

```

## ALGO-115. 和为T

### 问题描述

从一个大小为n的整数集中选取一些元素，使得它们的和等于给定的值T。每个元素限选一次，不能一个都不选。

### 输入格式

第一行一个正整数n，表示整数集内元素的个数。

第二行n个整数，用空格隔开。

第三行一个整数T，表示要达到的和。

输出格式

输出有若干行，每行输出一组解，即所选取的数字，按照输入中的顺序排列。

若有多组解，优先输出不包含第n个整数的；若都包含或都不包含，优先输出不包含第n-1个整数的，依次类推。

最后一行输出总方案数。

样例输入

```
5
-7 -3 -2 5 9
0
```

样例输出

```
-3 -2 5
-7 -2 9
2
```

数据规模和约定

$1 \leq n \leq 22$

$T \leq \text{maxlongint}$

集合中任意元素的和都不超过long的范围

分析：1.数据规模在 $n \leq 22$ ，递归搜索不会超时，每个数字可选择拿或不拿

2.搜索为了保证符合题目顺序，从后向前搜索～

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  vector<int> v, ans;
5  int n, k, cnt;
6  void dfs(int num, int sum) {
7      if (num == -1) {
8          if (sum == k && ans.size() > 0) {
9              for (int i = ans.size() - 1; i >= 0; i--) {
10                 if (i != 0) {
11                     printf("%d ", ans[i]);
12                 } else {
13                     printf("%d\n", ans[i]);
14                 }
15             }
16             cnt++;
17         }
18     } else {
19         dfs(num - 1, sum);
20         ans.push_back(v[num]);
```

```

21         dfs(num - 1, sum + v[num]);
22         ans.pop_back();
23     }
24 }
25 int main() {
26     scanf("%d", &n);
27     v.resize(n);
28     for (int i = 0; i < n; i++)
29         scanf("%d", &v[i]);
30     scanf("%d", &k);
31     dfs(n - 1, 0);
32     printf("%d\n", cnt);
33     return 0;
34 }

```

## ALGO-116. 最大的算式

### 问题描述

题目很简单，给出N个数字，不改变它们的相对位置，在中间加入K个乘号和N-K-1个加号，（括号随便加）使最终结果尽量大。因为乘号和加号一共就是N-1个了，所以恰好每两个相邻数字之间都有一个符号。例如：

N=5, K=2, 5个数字分别为1、2、3、4、5，可以加成：

$$1*2*(3+4+5)=24$$

$$1*(2+3)*(4+5)=45$$

$$(1*2+3)*(4+5)=45$$

.....

### 输入格式

输入文件共有二行，第一行为两个有空格隔开的整数，表示N和K，其中（ $2 \leq N \leq 15$ ,  $0 \leq K \leq N-1$ ）。第二行为 N 个用空格隔开的数字（每个数字在0到9之间）。

### 输出格式

输出文件仅一行包含一个整数，表示要求的最大的结果

### 样例输入

5 2

1 2 3 4 5

### 样例输出

120

### 样例说明

$$(1+2+3)*4*5=120$$

分析：用动态规划解决。设sum[i]为前i个数的总和，那么从j到k的总和为sum[k]-sum[j-1]。设dp[i][j]表示前i个数中有j个乘号的最大的结果。则要知道dp[i][j]，可以尝试从第二个数的前面一直到最后一个数的前面依次添加乘号，将最大的结果保存至dp[i][j]中。就可以得到状态转移方程为：dp[i][j] = max(dp[i][j], dp[l-1][j-1] \* (sum[i] - sum[l-1]));l为插入相乘的两个数的后一个数字的坐标。

```
1  #include <iostream>
2  using namespace std;
3  #define max(a, b) a > b ? a : b;
4  long long int dp[16][16];
5  int sum[16];
6
7  int main() {
8      int n, k;
9      cin >> n >> k;
10     for(int i = 1; i <= n; i++) {
11         int temp;
12         cin >> temp;
13         sum[i] = sum[i-1] + temp;
14     }
15     for(int i = 1; i <= n; i++) {
16         dp[i][0] = sum[i];
17     }
18     for(int i = 2; i <= n; i++) {
19         for(int j = 1; j <= i-1 && j <= k; j++) {
20             for(int l = 2; l <= n; l++) {
21                 dp[i][j] = max(dp[i][j], dp[l-1][j-1] * (sum[i] -
sum[l-1]));
22             }
23         }
24     }
25     cout << dp[n][k];
26     return 0;
27 }
```

## ALGO-117. 友好数

### 问题描述

有两个整数，如果每个整数的约数和（除了它本身以外）等于对方，我们就称这对数是友好的。例如：

9的约数和有：1+3=4

4的约数和有：1+2=3

所以9和4不是友好的。

220的约数和有：1 2 4 5 10 11 20 22 44 55 110=284

284的约数和有：1 2 4 71 142=220

所以220和284是友好的。

编写程序，判断两个数是否是友好数。

### 输入格式

一行，两个整数，由空格分隔

输出格式

如果是友好数，输出"yes"，否则输出"no"，注意不包含引号。

样例输入

220 284

样例输出

yes

数据规模和约定

两个整数都小于10000

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b, suma = 1, sumb = 1;
5      cin >> a >> b;
6      for (int i = 2; i * i <= a; i++) {
7          if (a % i == 0 && i * i == a) suma += i;
8          if (a % i == 0 && i * i != a) suma += i + a / i;
9      }
10     for (int i = 2; i * i <= b; i++) {
11         if (b % i == 0 && i * i == b) sumb += i;
12         if (b % i == 0 && i * i != b) sumb += i + b / i;
13     }
14     if (suma == b && sumb == a) cout << "yes";
15     else cout << "no";
16     return 0;
17 }
```

## ALGO-118. 连续正整数的和

问题描述

78这个数可以表示为连续正整数的和， $1+2+3$ ， $18+19+20+21$ ， $25+26+27$ 。

输入一个正整数  $n$  ( $n \leq 10000$ )

输出  $m$  行 ( $n$  有  $m$  种表示法)，每行是两个正整数  $a, b$ ，表示  $a+(a+1)+\dots+b=n$ 。

对于多种表示法， $a$  小的方案先输出。

样例输入

78

样例输出

1 12

18 21

25 27

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int *a = new int [n+1];
7      a[0] = 0;
8      for(int i = 1; i <= n; i++) {
9          a[i] = a[i-1] + i;
10     }
11     for(int i = 0; i <= n-2; i++) {
12         for(int j = i+1; j <= n; j++) {
13             if((a[j] - a[i]) == n) {
14                 cout << i+1 << " " << j << endl;
15             } else if((a[j] - a[i]) > n) {
16                 break;
17             }
18         }
19     }
20     delete [] a;
21     return 0;
22 }
```

## ALGO-119. 寂寞的数

### 问题描述

道德经曰：一生二，二生三，三生万物。

对于任意正整数 $n$ ，我们定义 $d(n)$ 的值为为 $n$ 加上组成 $n$ 的各个数字的和。例如， $d(23)=23+2+3=28$ ,  $d(1481)=1481+1+4+8+1=1495$ 。

因此，给定了任意一个 $n$ 作为起点，你可以构造如下一个递增序列： $n, d(n), d(d(n)), d(d(d(n)))...$ 例如，从33开始的递增序列为：

33, 39, 51, 57, 69, 84, 96, 111, 114, 120, 123, 129, 141, ...

我们把 $n$ 叫做 $d(n)$ 的生成元，在上面的数列中，33是39的生成元，39是51的生成元，等等。有一些数字甚至可以有二个生成元，比如101，可以由91和100生成。但也有一些数字没有任何生成元，如42。我们把这样的数字称为寂寞的数字。

### 输入格式

一行，一个正整数 $n$ 。

### 输出格式

按照升序输出小于 $n$ 的所有寂寞的数字，每行一个。



样例输入

40

样例输出

1

3

5

7

9

20

31

数据规模和约定

$n \leq 10000$

```
1  #include <iostream>
2  using namespace std;
3  int a[10001];
4  int main() {
5      int n;
6      cin >> n;
7      for(int i = 1; i <= n; i++) {
8          int temp = i;
9          int num = i;
10         while(num) {
11             temp += num%10;
12             num = num/10;
13         }
14         a[temp] = 1;
15     }
16     for(int i = 1; i <= n; i++) {
17         if(a[i] == 0)
18             cout << i << endl;
19     }
20     return 0;
21 }
```

## ALGO-120. 学做菜

问题描述

涛涛立志要做新好青年，他最近在学做菜。由于技术还很生疏，他只会用鸡蛋，西红柿，鸡丁，辣酱这四种原料来做菜，我们给这四种原料标上字母A,B,C,D。

涛涛现在会做的菜有五种：

1、西红柿炒鸡蛋 原料：AABDD

- 2、酸辣鸡丁 原料: ABCD
- 3、宫保鸡丁 原料: CCD
- 4、水煮西红柿 原料: BBB
- 5、怪味蛋 原料: AD

这天早上, 开开去早市给涛涛买了一些原料回来。由于事先没有什么计划, 涛涛决定, 对于现存的原料, 每次尽量做菜单上靠前 (即编号小) 的菜。

现在请你写一个程序, 判断一下开开和涛涛中午能吃到哪些菜。

输入格式

共4个整数a,b,c,d。分别表示开开买的A,B,C,D这4种原料的数量。每种原料不会超过30份。

输出格式

输出5行。其中第i行表示涛涛做的第i种菜的数目。

样例输入

3  
1  
2  
4

样例输出

1  
0  
1  
0  
1

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int a, b, c, d;
6      cin >> a >> b >> c >> d;
7      if (a >= 2 && b >= 1 && d >= 2) {
8          int minn1 = min(a / 2, b / 1);
9          int minn = min(minn1, d / 2);
10         a -= minn * 2;
11         b -= minn * 1;
12         d -= minn * 2;
13         cout << minn << endl;
14     }else{
15         cout << 0 << endl;
16     }
17     if (a >= 1 && b >= 1 && d >= 1) {
18         int minn1 = min(a / 1, b / 1);
19         int minn2 = min(c / 1, d / 1);
20         int minn = min(minn1, minn2);
```

```

21     a -= minn * 1;
22     b -= minn * 1;
23     c -= minn * 1;
24     d -= minn * 1;
25     cout << minn << endl;
26 }else{
27     cout << 0 << endl;
28 }
29 if (c >= 2 && d >= 1) {
30     int minn = min(c / 2, d / 1);
31     c -= minn * 2;
32     d -= minn * 1;
33     cout << minn << endl;
34 }else{
35     cout << 0 << endl;
36 }
37 if (b >= 3) {
38     int minn = b / 3;
39     c -= minn * 3;
40     cout << minn << endl;
41 }else{
42     cout << 0 << endl;
43 }
44 if(a >= 1 && d >= 1){
45     int minn = min(a / 1, d / 1);
46     cout << minn << endl;
47 }else{
48     cout << 0 << endl;
49 }
50 return 0;
51 }

```

## ALGO-121. 猴子分苹果

### 问题描述

秋天到了， $n$ 只猴子采摘了一大堆苹果放到山洞里，约定第二天平分。这些猴子很崇拜猴王孙悟空，所以都想给他留一些苹果。第一只猴子悄悄来到山洞，把苹果平均分成 $n$ 份，把剩下的 $m$ 个苹果吃了，然后藏起来一份，最后把剩下的苹果重新合在一起。这些猴子依次悄悄来到山洞，都做同样的操作，恰好每次都剩下了 $m$ 个苹果。第二天，这些猴子来到山洞，把剩下的苹果分成 $n$ 分，巧了，还是剩下了 $m$ 个。问，原来这些猴子至少采了多少个苹果。

### 输入格式

两个整数， $n$   $m$

### 输出格式

一个整数，表示原来苹果的数目

### 样例输入

5 1

样例输出

15621

数据规模和约定

$0 < m < n < 9$

分析：1.每次猴子分完桃子后剩下的桃子一定是分之前的 $(n-1)/n$ ，所以每次分剩的桃子是 $n-1$ 的倍数  
2.枚举，大家最后一起吃桃子时，每人最终还有几个（从0开始，0是边界数据），保证每次分后的数量是4的倍数即可， $pre = now / 4 * 5 + m$ ,倒推 $n$ 次即可~

```
1  #include <iostream>
2  using namespace std;
3  int m, n;
4  int check(int num) {
5      for (int i = 0; i < n; i++) {
6          if (num % (n-1) == 0) num = num / (n-1) * n + m;
7          else return 0;
8      }
9      return num;
10 }
11 int main() {
12     cin >> n >> m;
13     for (int i = 0;; i++) {
14         int num = i * n + m;
15         if (check(num)) {
16             cout << check(num);
17             return 0;
18         }
19     }
20 }
```

## ALGO-122. 未名湖边的烦恼

问题描述

每年冬天，北大未名湖上都是滑冰的好地方。北大体育组准备了许多冰鞋，可是人太多了，每天下午收工后，常常一双冰鞋都不剩。

每天早上，租鞋窗口都会排起长龙，假设有还鞋的 $m$ 个，有需要租鞋的 $n$ 个。现在的问题是，这些人有多少种排法，可以避免出现体育组没有冰鞋可租的尴尬场面。（两个同样需求的人（比如都是租鞋或都是还鞋）交换位置是同一种排法）

输入格式

两个整数，表示 $m$ 和 $n$

输出格式

一个整数，表示队伍的排法的方案数。

样例输入

3 2

样例输出

5

数据规模和约定

$m, n \in [0, 18]$

```
1  #include <iostream>
2  using namespace std;
3  int f(int m, int n) {
4      if (m < n)
5          return 0;
6      if (n == 0)
7          return 1;
8      return f(m - 1, n) + f(m, n - 1);
9  }
10
11 int main() {
12     int m, n;
13     cin >> m >> n;
14     cout << f(m, n);
15     return 0;
16 }
```

## ALGO-123. A+B problem

问题描述

Given two integers A and B, your task is to output their sum, A+B.

输入格式

The input contains of only one line, consisting of two integers A and B. ( $0 \leq A, B \leq 1\,000$ )

输出格式

The output should contain only one number that is A+B.

样例输入

1 1

样例输出

2

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b;
5      cin >> a >> b;
6      cout << a + b;
7      return 0;
8  }

```

## ALGO-124. 数字三角形

### 问题描述

(图 3.1 - 1) 示出了一个数字三角形。请编一个程序计算从顶至底的某处的一条路径，使该路径所经过的数字的总和最大。

- 每一步可沿左斜线向下或右斜线向下走；
- $1 < \text{三角形行数} \leq 100$ ；
- 三角形中的数字为整数 0, 1, ..., 99；

```

      7
     3 8
    8 1 0
   2 7 4 4
  4 5 2 6 5

```

(图 3.1 - 1)

### 输入格式

文件中首先读到的是三角形的行数。

接下来描述整个三角形

### 输出格式

最大总和 (整数)

### 样例输入

```

5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5

```

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstring>
4  int f(int i, int j);
5  int a[101][101];
6  int d[101][101];
7  int n;
8  using namespace std;
9  int main() {
10     memset(d, -1, sizeof(d));
11     cin >> n;
12     for (int i = 1; i <= n; i++)
13         for (int j = 1; j <= i; j++)
14             cin >> a[i][j];
15     cout << f(1, 1);
16     return 0;
17 }
18
19 int f(int i, int j) {
20     if (d[i][j] >= 0)
21         return d[i][j];
22     if (i == n)
23         return a[i][j];
24     else
25         return d[i][j] = a[i][j] + max(f(i + 1, j), f(i + 1, j + 1));
26 }

```

## ALGO-126. 水仙花

### 问题描述

判断给定的三位数是否 水仙花 数。所谓 水仙花 数是指其值等于它本身 每位数字立方和的数。例 153 就是一个 水仙花 数。153=13+53+33

### 输入格式

一个整数。

### 输出格式

是水仙花数,输出"YES",否则输出"NO"(不包括引号)

### 样例输入

123

### 样例输出

NO

一个三位的整数,否则输出"NO"

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string s;
6      cin >> s;
7      int a = 0;
8      int sum = 0;
9      for(int i = 0; i < s.length(); i++) {
10         int temp = s[i] - '0';
11         a = a * 10 + temp;
12         sum += (temp * temp * temp);
13     }
14     if(sum == a) {
15         cout << "YES";
16     } else {
17         cout << "NO";
18     }
19     return 0;
20 }
```

## ALGO-129. 特殊的数字四十

### 问题描述

1234是一个非常特殊的四位数,因为它的各位数之和为10,编程求所有这样的四位十进制数。

### 输出格式

按从小到大的顺序输出满足条件的四位十进制数。每个数字占用一行。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      for(int i = 1; i <= 9; i++) {
5          for(int j = 0; j <= 9; j++) {
6              for(int k = 0; k <= 9; k++) {
7                  int l = 10 - i - j - k;
8                  if(l >= 0 && l <= 9)
9                      cout << i << j << k << l << endl;
10             }
11         }
12     }
13     return 0;
14 }
```



## ALGO-139. s01串

### 问题描述

s01串初始为"0"

按以下方式变换

0变1, 1变01

### 输入格式

1个整数(0~19)

### 输出格式

n次变换后s01串

### 样例输入

3

### 样例输出

101

### 数据规模和约定

0~19

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      int n;
6      cin >> n;
7      string s = "0";
8      while(n-->0) {
9          for(int i = 0; i < s.length(); i++) {
10             if(s[i] == '0') {
11                 s[i] = '1';
12             } else {
13                 s.insert(i, "0");
14                 i = i+1;
15             }
16         }
17     }
18     cout << s;
19     return 0;
20 }
```

## ALGO-140. P1101

有一份提货单，其数据项目有：商品名（MC）、单价（DJ）、数量（SL）。定义一个结构体prut，其成员是上面的三项数据。在主函数中定义一个prut类型的结构体数组，输入每个元素的值，计算并输出提货单的总金额。

输入格式：第一行是数据项个数N(N<100)，接下来每一行是一个数据项。商品名是长度不超过100的字符串，单价为double类型，数量为整型。

输出格式：double类型的总金额。

输入：

```
4
book 12.5 3
pen 2.5 10
computer 3200 1
flower 47 5
```

输出：

```
3497.500000
```

```
1  #include <iostream>
2  #include <cstdio>
3  #include <vector>
4  using namespace std;
5  struct prut{
6      string MC;
7      double DJ;
8      int SL;
9  };
10 int main() {
11     int N;
12     cin >> N;
13     double sum = 0.0;
14     vector<prut> a(N);
15     for(int i = 0; i < N; i++) {
16         cin >> a[i].MC >> a[i].DJ >> a[i].SL;
17         sum += a[i].DJ * a[i].SL;
18     }
19     printf("%lf", sum);
20     return 0;
21 }
```

## ALGO-141. P1102

定义一个学生结构体类型student，包括4个字段，姓名、性别、年龄和成绩。然后在主函数中定义一个结构体数组（长度不超过1000），并输入每个元素的值，程序使用冒泡排序法将学生按照成绩从小到大的顺序排序，然后输出排序的结果。

输入格式：第一行是一个整数N（N<1000），表示元素个数；接下来N行每行描述一个元素，姓名、性别都是长度不超过20的字符串，年龄和成绩都是整型。

输出格式：按成绩从小到大输出所有元素，若多个学生成绩相同则成绩相同的同学之间保留原来的输入顺序。

输入：

3

Alice female 18 98

Bob male 19 90

Miller male 17 92

输出：

Bob male 19 90

Miller male 17 92

Alice female 18 98

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  struct student {
5      string name;
6      string sex;
7      int age;
8      int score;
9  };
10
11 int main() {
12     int n;
13     cin >> n;
14     vector<student> a(n);
15     for (int i = 0; i < n; i++)
16         cin >> a[i].name >> a[i].sex >> a[i].age >> a[i].score;
17     for (int i = 0; i < n - 1; i++) {
18         for(int j = 0; j < n - 1 - i; j++) {
19             if(a[j].score > a[j+1].score)
20                 swap(a[j], a[j+1]);
21         }
22     }
23     for (int i = 0; i < n; i++)
24         cout << a[i].name << " " << a[i].sex << " " << a[i].age << " "
25         << a[i].score << endl;
26     return 0;
27 }
```

## ALGO-142. P1103

编程实现两个复数的运算。设有两个复数  $z_1$  和  $z_2$ ，则他们的运算公式为：

要求：

- (1) 定义一个结构体类型来描述复数。
- (2) 复数之间的加法、减法、乘法和除法分别用不同的函数来实现。
- (3) 必须使用结构体指针的方法把函数的计算结果返回。

说明：

用户输入：运算符(+,-,\*,/) a b c d.

输出：

a+bi，输出时不管a,b是小于0或等于0都按该格式输出，输出时a,b都保留两位。

输入：

- 2.5 3.6 1.5 4.9

输出：

1.00+-1.30i

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  struct node {
5      float a;
6      float b;
7  };
8
9  char t;
10
11 struct node m, n, p, *q;
12
13 struct node * add() {
14     p.a = m.a + n.a;
15     p.b = m.b + n.b;
16     q = &p;
17     return q;
18 }
19
20 struct node * minu() {
21     p.a = m.a - n.a;
22     p.b = m.b - n.b;
23     q = &p;
24     return q;
25 }
26
27 struct node * multi() {
28     p.a = m.a * n.a - m.b * n.b;
```

```

29     p.b = m.a * n.b + m.b * n.a;
30     q = &p;
31     return q;
32 }
33
34 struct node * div() {
35     p.a = (m.a * n.a + m.b * n.b) / (n.a * n.a + n.b * n.b);
36     p.b = (m.b * n.a - m.a * n.b) / (n.a * n.a + n.b * n.b);
37     q = &p;
38     return q;
39 }
40
41 int main() {
42     scanf("%c %f %f %f %f", &t, &m.a, &m.b, &n.a, &n.b);
43     if(t == '+')
44         add();
45     if(t == '-')
46         minu();
47     if(t == '*')
48         multi();
49     if(t == '/')
50         div();
51     printf("%.2f+%.2fi", q->a, q->b);
52     return 0;
53 }

```

## ALGO-143. 字符串变换

### 问题描述

相信经过这个学期的编程训练，大家对于字符串的操作已经掌握的相当熟练了。今天，徐老师想测试一下大家对于字符串操作的掌握情况。徐老师自己定义了1,2,3,4,5这5个参数分别指代不同的5种字符串操作，你需要根据传入的参数，按照徐老师的规定，对输入字符串进行格式转化。

徐老师指定的操作如下：

- 1 表示全部转化为大写字母输出，如abC 变成 ABC
- 2 表示全部转换为小写字母输出，如abC变成abc
- 3 表示将字符串整个逆序输出，如 abc 变成 cba
- 4 表示将字符串中对应的大写字母转换为小写字母，而将其中的小写字母转化为大写字母输出，如 abC变成ABc
- 5 表示将全部转换为小写字母，并将其中所有的连续子串转换为对应的缩写形式输出，比如abcD转换为a-d，其次，-至少代表1个字母，既如果是ab，则不需要转换为缩写形式。

### 输入格式

一共一行，分别是指代对应操作的数字和字符串，两者以空格分隔，字符串全部由英文字母组成

### 输出格式

输出根据上述规则转换后对应的字符串

样例输入

5 ABcdEE

样例输出

a-ee

数据规模和约定

输入字符串长度最长为200。

```
1  #include <iostream>
2  #include <cctype>
3  using namespace std;
4  int main() {
5      int n;
6      cin >> n;
7      string s;
8      cin >> s;
9      int len = s.length();
10     switch(n) {
11         case 1:
12         case 2:
13         case 4:
14             for(int i = 0; i < len; i++) {
15                 if(n == 1)
16                     s[i] = toupper(s[i]);
17                 else if(n == 2)
18                     s[i] = tolower(s[i]);
19                 else {
20                     if(isupper(s[i]))
21                         s[i] = tolower(s[i]);
22                     else
23                         s[i] = toupper(s[i]);
24                 }
25             }
26             cout << s;
27             break;
28
29         case 3:
30             for(int i = len-1; i >= 0; i--) {
31                 cout << s[i];
32             }
33             break;
34
35         case 5:
36             string t;
37             for(int i = 0; i < len; i++) {
```

```

38         s[i] = tolower(s[i]);
39     }
40     for(int i = 0; i < len; i++) {
41         if(i == 0) {
42             t += s[0];
43         } else if(i != len-1) {
44             if(s[i] != s[i-1]+1 || s[i] != s[i+1]-1) {
45                 t += s[i];
46             } else if(t[t.length()-1] != '-') {
47                 t += '-';
48             }
49         } else {
50             t += s[i];
51         }
52     }
53     cout << t;
54     break;
55 }
56 return 0;
57 }

```

## ALGO-145. 4-1打印下述图形

### 问题描述

使用循环结构打印下述图形，打印行数n由用户输入。打印空格时使用"%s"格式，向printf函数传递只包含一个或多个空格的字符串" ",下同。

### 样例输入

一个满足题目要求的输入范例。

5

### 样例输出

与上面的样例输入对应的输出。

```

      *
     ***
    *****
   *********
  ***********
 *****

```

### 数据规模和约定

输入数据中每一个数的范围。

例：0<n<20。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      for(int i = 0; i < n; i++) {
7          for(int j = 0; j < n-i-1; j++)
8              cout << " ";
9          for(int j = 0; j < 2*i+1; j++)
10             cout << "*";
11         cout << endl;
12     }
13     return 0;
14 }

```

## ALGO-147. 4-3水仙花数

### 问题描述

打印所有100至999之间的水仙花数。所谓水仙花数是指满足其各位数字立方和为该数字本身的整数，例如  $153=1^3+5^3+3^3$ 。

### 样例输入

一个满足题目要求的输入范例。

例：

无

### 样例输出

153

xxx

xxx

```

1  #include <stdio>
2  using namespace std;
3  int main() {
4      for(int i = 1; i <= 9; i++) {
5          for(int j = 0; j <= 9; j++) {
6              for(int k = 0; k <= 9; k++) {
7                  int sum = i * i * i + j * j * j + k * k * k;
8                  int n = i * 100 + j * 10 + k;
9                  if(sum == n)
10                     printf("%d%d%d\n", i, j, k);
11             }
12         }
13     }
14     return 0;

```



## ALGO-148. 5-1最小公倍数

### 问题描述

编写一函数lcm，求两个正整数的最小公倍数。

### 样例输入

一个满足题目要求的输入范例。

例：

```
3 5
15请按任意键继续. . .
```

35

### 样例输出

与上面的样例输入对应的输出。

例：

15

### 数据规模和约定

输入数据中每一个数的范围。

例：两个数都小于65536。

分析：辗转相除法求最大公约数，乘积除以最大公约数等于最小公倍数～

```
1  #include <iostream>
2  using namespace std;
3  int gcd(int a, int b) {
4      return b == 0 ? a : gcd(b, a%b);
5  }
6  int main() {
7      int a, b;
8      cin >> a >> b;
9      cout << a * b / gcd(a, b);
10     return 0;
11 }
```

## ALGO-149. 5-2求指数

### 问题描述

已知n和m，打印 $n^1$ ， $n^2$ ，...， $n^m$ 。要求用静态变量实现。 $n^m$ 表示n的m次方。已知n和m，打印 $n^1$ ， $n^2$ ，...， $n^m$ 。要求用静态变量实现。 $n^m$ 表示n的m次方。（每行显示5个数，每个数宽为12，右对齐）

样例输入

一个满足题目要求的输入范例。

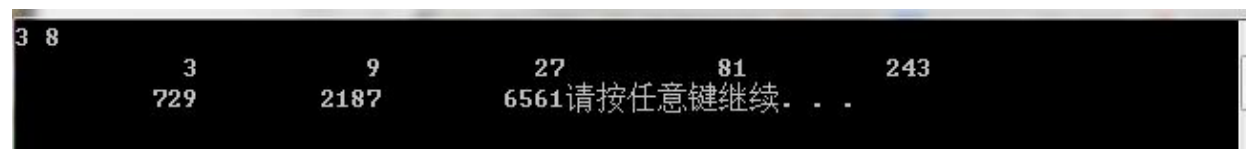
例：

3 8

样例输出

与上面的样例输入对应的输出。

例：



```
3 8
      3      9      27      81      243
    729    2187    6561 请按任意键继续. . .
```

数据规模和约定

输入数据中每一个数的范围。

例： $n^m$ 小于int 的表示范围。

分析：用静态变量表示result和m，每次将m减一，result累乘并输出，以%12d的形式输出～

```
1  #include <stdio>
2  static int result = 1, m;
3  int n, rest, flag = 0;
4  void func() {
5      result *= n;
6      if (flag == 1 && (m - rest) % 5 == 0)
7          printf("\n");
8      flag = 1;
9      m--;
10     printf("%12d", result);
11 }
12 int main() {
13     scanf("%d%d", &n, &m);
14     rest = m % 5;
15     while (m != 0)
16         func();
17     return 0;
18 }
```

## ALGO-149. 5-2求指数

问题描述

已知n和m，打印 $n^1, n^2, \dots, n^m$ 。要求用静态变量实现。 $n^m$ 表示n的m次方。已知n和m，打印 $n^1, n^2, \dots, n^m$ 。要求用静态变量实现。 $n^m$ 表示n的m次方。（每行显示5个数，每个数宽为12，右对齐）

样例输入

一个满足题目要求的输入范例。

例：

3 8

样例输出

与上面的样例输入对应的输出。

例：

数据规模和约定

输入数据中每一个数的范围。

例： $n^m$ 小于int 的表示范围。

分析：用静态变量表示result和m，每次将m减一，result累乘并输出，以%12d的形式输出～

```
1  #include <stdio>
2  static int result = 1, m;
3  int n, rest, flag = 0;
4  void func() {
5      result *= n;
6      if (flag == 1 && (m - rest) % 5 == 0)
7          printf("\n");
8      flag = 1;
9      m--;
10     printf("%12d", result);
11 }
12 int main() {
13     scanf("%d%d", &n, &m);
14     rest = m % 5;
15     while (m != 0)
16         func();
17     return 0;
18 }
```

## ALGO-150. 6-1 递归求二项式系数值

问题描述

已知 $C_n^k = \begin{cases} 1 & k = 0 \text{ 或 } k = n \\ C_{n-1}^k + C_{n-1}^{k-1} & 0 < k < n \end{cases}$ ，使用递归方法求 $C_n^k$ 。

样例输入

一个满足题目要求的输入范例。

3 10

样例输出

与上面的样例输入对应的输出。

```
3 10
120请按任意键继续. . .
```

数据规模和约定

输入数据中每一个数的范围。

例：结果在int表示时不会溢出。

```
1  #include <iostream>
2  using namespace std;
3  int dfs(int n, int k) {
4      if (k == 0 || k == n) return 1;
5      return dfs(n-1, k) + dfs(n-1, k-1);
6  }
7  int main() {
8      int k, n;
9      cin >> k >> n;
10     cout << dfs(n, k);
11     return 0;
12 }
```

## ALGO-151. 6-2递归求二进制表示位数

问题描述

给定一个十进制整数，返回其对应的二进制数的位数。例如，输入十进制数9，其对应的二进制数是1001，因此位数是4。

样例输入

9

样例输出

4

数据规模和约定

输入数据中每一个数的范围。

例：输入在int表示范围内。

分析：利用二进制移位计算，每次向右移动一位，直到为0为止，返回移位的次数cnt~

```

1  #include <iostream>
2  using namespace std;
3  int dfs(int n, int cnt) {
4      if (n == 0) return cnt;
5      return dfs(n >> 1, cnt + 1);
6  }
7  int main() {
8      int n;
9      cin >> n;
10     cout << dfs(n, 0);
11     return 0;
12 }

```

## ALGO-152. 8-2求完数

### 问题描述

如果一个自然数的所有小于自身的因子之和等于该数，则称为完数。设计算法，打印1-9999之间的所有完数。

### 样例输出

与上面的样例输入对应的输出。

例：6就是一个完数

### 数据规模和约定

1-9999

分析：求6~9999之间所有的数的因子之和，然后将和sum与自身比较，如果相等就输出～

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      for (int i = 6; i <= 9999; i++) {
5          int sum = 1;
6          for (int j = 2; j < i; j++) {
7              if (i % j == 0)
8                  sum += j;
9          }
10         if (sum == i)
11             cout << i << endl;
12     }
13     return 0;
14 }

```

## ALGO-157. 阶乘末尾

### 问题描述

给定n和len，输出n!末尾len位。

输入格式

一行两个正整数n和len。

输出格式

一行一个字符串，表示答案。长度不足用前置零补全。

样例输入

6 5

样例输出

00720

数据规模和约定

$n \leq 30, \text{len} \leq 10$ 。

分析：1.每次只要取后len位的数字就够了，输出的时候位数不够要补0

2.判断数字位数的时候可以取以10为底的对数（蓝桥杯不支持c++ 11中的to\_string()方法哦~）

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int n, len, cnt = 0;
6      cin >> n >> len;
7      long long res, ans = 1, m = pow(10, len);
8      for (int i = 1; i <= n; i++)
9          ans = (ans * i) % m;
10     if (ans == 0) cnt = 1;
11     else cnt = log10(ans) + 1;
12     string s(len - cnt, '0');
13     printf("%s%lld", s.c_str(), ans);
14     return 0;
15 }
```

## ALGO-158. sign函数

问题描述

给定实数x，输出sign(x)的值。

sign(x)是符号函数，如果 $x > 0$ ，则返回1；如果 $x = 0$ ，则返回0；如果 $x < 0$ ，则返回-1。

输入格式

一行一个实数x。

输出格式

一行一个整数表示答案。

样例输入

-0.0001

样例输出

-1

数据规模和约定

$|x| \leq 10000$ ，输入数据精度最多达到4位小数。

提示

判断实数 $x$ 是否等于零时，由于计算机实数运算误差，应当引入极小量 $\epsilon$ ，核心代码如下：  
其中 $\text{fabs}$ 为 $\text{cmath}$ 中的绝对值函数。

```
const double eps=1e-6;
```

```
if (fabs(x) <= eps) {
```

```
//x是零
```

```
}
```

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      double n;
5      cin >> n;
6      if(n > 0) cout << 1;
7      else if(n < 0) cout << -1;
8      else if(n == 0) cout << 0;
9      return 0;
10 }
```

## ALGO-159. P0103

从键盘输入一个大写字母，要求改用小写字母输出。

输入

A

输出

a

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      char c;
5      cin >> c;
6      cout << (char) (c + 32);
7      return 0;
8  }
```

## ALGO-160. P0104

求方程 $ax^2+bx+c=0$ 的实数根。a, b, c由键盘输入,  $a \neq 0$ 。若只有一个实数根 ( $b^2-4ac=0$ ) 则只输出x1, 若无实数根 ( $b^2-4ac<0$ ) 则输出Error。

输入

2.5 7.5 1.0

输出

(注意等号前面后面都有一个空格)

x1 = -0.139853

x2 = -2.860147

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      double a , b, c;
6      cin >> a >> b >> c;
7      double det = b * b - 4 * a * c;
8      if(det < 0) {
9          printf("%s", "Error\n");
10     }else if(det == 0 ) {
11         printf("x1 = %6f\n", -1 * b / (2 * a));
12     }else {
13         printf("x1 = %6f\n", (-1 * b + sqrt(det)) / (2 * a));
14         printf("x2 = %6f\n", (-1 * b - sqrt(det)) / (2 * a));
15     }
16     return 0;
17 }
```

## ADV-9. 递归倒置字符数组

问题描述

完成一个递归程序，倒置字符数组。并打印实现过程

递归逻辑为：

当字符长度等于1时，直接返回

否则，调换首尾两个字符，在递归地倒置字符数组的剩下部分

输入格式

字符数组长度及该数组

输出格式

在求解过程中，打印字符数组的变化情况。

最后空一行，在程序结尾处打印倒置后该数组的各个元素。

样例输入



Sample 1

5 abcde

Sample 2

1 a

样例输出

Sample 1

ebcda

edcba

edcba

Sample 2

a

分析：测试用例坑坑哒。。最后会多输出一遍倒置完了的字符串就算了。。还会在前面多加一个空行。。。 (手动再见--||)

```
1  #include <iostream>
2  using namespace std;
3  string s;
4  void func(int head, int tail) {
5      if(tail - head <= 0)
6          return ;
7      swap(s[head], s[tail]);
8      cout << s << endl;
9      func(head+1, tail-1);
10 }
11
12 int main() {
13     int n;
14     cin >> n >> s;
15     func(0, n-1);
16     cout << endl << s << endl;
17     return 0;
18 }
```

## ADV-11. Torry的困惑(提高型)

问题描述

Torry从小喜爱数学。一天，老师告诉他，像2、3、5、7.....这样的数叫做质数。Torry突然想到一个问题，前10、100、1000、10000.....个质数的乘积是多少呢？他把这个问题告诉老师。老师愣住了，一时回答不出来。于是Torry求助于会编程的你，请你算出前n个质数的乘积。不过，考虑到你才接触编程不久，Torry只要你算出这个数模上50000的值。

输入格式

仅包含一个正整数n，其中n<=100000。

输出格式

输出一行，即前n个质数的乘积模50000的值。

样例输入

1

样例输出

2

分析：和leetcode里面那道easy的题目-LeetCode 204. Count Primes方法一样~用v[i]数组标记当前是否为质数，先初始化为都为质数==0，然后只需去除从2到根号n的，从i\*i开始的所有i的倍数即可~当然，当当前v[i]已经标记为不是质数的时候，就无需判断它的倍数了，因为例如16是4的倍数的同时，如果已知4是2的倍数，那么16一定是2的倍数~~

```
1  #include <iostream>
2  #define MOD 50000
3  using namespace std;
4  int v[2000000];
5  int main() {
6      int n;
7      cin >> n;
8      for(int i = 2; i * i < 2000000; i++) {
9          if(v[i] == 1)
10             continue;
11             for(int j = i * i; j < 2000000; j = j + i)
12                 v[j] = 1;
13     }
14     long long int ans = 1;
15     int cnt = 0;
16     for(int i = 2; i < 2000000; i++) {
17         if(v[i] == 0) {
18             ans = (ans * i) % MOD;
19             cnt++;
20         }
21         if(cnt == n)
22             break;
23     }
24     cout << ans;
25     return 0;
26 }
```

## ADV-12. 计算时间

问题描述

给定一个t，将t秒转化为HH:MM:SS的形式，表示HH小时MM分钟SS秒。HH,MM,SS均是两位数，如果小于10用0补到两位。

输入格式

第一行一个数T( $1 \leq T \leq 100,000$ ), 表示数据组数。后面每组数据读入一个数t,  $0 \leq t < 24 \times 60 \times 60$ 。

输出格式

每组数据一行, HH:MM:SS。

样例输入

2

0

86399

样例输出

00:00:00

23:59:59

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int k, n;
5      cin >> k;
6      for (int i = 0; i < k; i++) {
7          cin >> n;
8          int s = n % 60;
9          int m = (n / 60) % 60;
10         int h = n / (60 * 60);
11         printf("%02d:%02d:%02d\n", h, m, s);
12     }
13     return 0;
14 }
```

### ADV-13. 最小乘积(提高型)

问题描述

给两组数, 各n个。

请调整每组数的排列顺序, 使得两组数据相同下标元素对应相乘, 然后相加的和最小。要求程序输出这个最小值。

例如两组数分别为:1 3 -5和-2 4 1

那么对应乘积取和的最小值应为:

$(-5) \times 4 + 3 \times (-2) + 1 \times 1 = -25$

输入格式

第一个行一个数T表示数据组数。后面每组数据, 先读入一个n, 接下来两行每行n个数, 每个数的绝对值小于等于1000。

$n \leq 1000, T \leq 10$

输出格式

一个数表示答案。

样例输入

```
2
3
1 3 -5
-2 4 1
5
1 2 3 4 5
1 0 1 0 1
```

样例输出

```
-25
6
```

分析：两组数据，一组升序排列，另一组降序排列，对应下标的数字相乘，最后求和～

```
1  #include <map>
2  #include <string>
3  using namespace std;
4  string add(string s1, string s2) {
5      int len1 = s1.length(), len2 = s2.length();
6      if (len1 < len2) {
7          string t(len2 - len1, '0');
8          s1 = t + s1;
9      } else if (len2 < len1) {
10         string t(len1 - len2, '0');
11         s2 = t + s2;
12     }
13     string ans = s1;
14     int car = 0;
15     for (int i = s1.length() - 1; i >= 0; i--) {
16         ans[i] = (s1[i] - '0' + s2[i] - '0' + car) % 10 + '0';
17         car = (s1[i] - '0' + s2[i] - '0' + car) / 10;
18     }
19     if (car) ans = (char) (car + '0') + ans;
20     return ans;
21 }
22 int main() {
23     string a, b;
24     cin >> a >> b;
25     cout << add(a, b);
26     return 0;
27 }
```

## ADV-14. 卡勒沃夫之弱水路三千(提高型)

问题描述

锦瑟年华谁与度 莫问情归处 只影向斜阳 剑吼西风 欲把春留驻  
天涯芳草无归路 回首花无数 解语自销魂 弱袂萦春 尘缘不相误

.....

在卡勒沃夫充满文学杀伤力的声音中，身处紫荆2号楼202B的四位远近高低各不相同的室友纷纷回忆起了各自波澜起伏的过去，并对长在百草园，邻有百花谷的现状表达了各自的见解。

某Q：“...我小学就开窍了...她的父母说我很好，但是...今天又和北林的联系了...”

某X：“...差点就成了，结果到学校了...这个方法放假了我去对我的同桌用！...”

某W：“...”（千言万语不言中，有大量的故事等待考古）

某Z：“...为了来清华...咱们审美观不一样，不会抢...”

.....

卡勒沃夫在这个不朽的夜话中搜集出了某人零散的历任女友资料，为了强迫某人将他出的题目的标程交出，现在卡勒沃夫需要一个能将这些零散信息整合起来的程序。伴随着雄壮委婉动人的音乐，身为程序设计快男（超女）的你降临了！卡勒沃夫正对着您做Orz状并请求着：“神牛啊~请施舍给我一段程序把~偶米头发~”。。。

#### 输入格式

第一行为一个不超过5的整数T，表示数据的组数。之后每组数据的一行为一个不超过100的整数n。之后n行每行有两个用单个空格隔开的字符串（每个字符串只有英文大小写字母，长度不超过10），为两位mm的名字。每行第一个mm先于第二个mm成为某人的女友。

在这里我们假装诅咒某人不会同时被两个或两个以上mm泡，某个mm抛弃了某人后不会再吃回头草，同时卡勒沃夫深邃的洞察力使得他收集到了充足的信息以确定某人女友的先后顺序。

在小数据组中出现的人物不超过13个

#### 输出格式

输出T行，每行对应一组数据，并按照mm们从先到后成为某人女友的顺序输出她们的名字，各个名字间用一个空格隔开。

#### 样例输入

```
2
2
RY Unknown
YSZ RY
3
tomorrow yestoday
tomorrow today
today yestoday
```

#### 样例输出

```
YSZ RY Unknown
tomorrow today yestoday
```

分析：1.用girl映射到一组girl,为某个女孩映射到前面有哪些女孩

2.如果某个女孩前面没有女孩，并且没有被输出过，则输出当前女孩并标记，并且，将其从其他所有女孩的映射中擦除

3.重复2过程，直到所有女孩被输出~

```
1 #include <iostream>
```

```

2  #include <algorithm>
3  #include <set>
4  #include <map>
5  using namespace std;
6  int main() {
7      int k, n;
8      cin >> k;
9      while (k--) {
10         string ans = "";
11         cin >> n;
12         map<string, set<string> > m;
13         map<string, int> vis;
14         for (int i = 0; i < n; i++) {
15             string s1, s2;
16             cin >> s1 >> s2;
17             m[s1];
18             m[s2].insert(s1);
19         }
20         while (1) {
21             int check = 0;
22             for (map<string, set<string> >::iterator i = m.begin(); i
!= m.end(); i++) {
23                 string girl = i->first;
24                 if (vis[girl] == 0 && m[girl].size() == 0) {
25                     check = 1;
26                     vis[girl] = 1;
27                     for (map<string, set<string> >::iterator j =
m.begin(); j != m.end(); j++) {
28                         j->second.erase(girl);
29                     }
30                     ans = ans + girl + " ";
31                     break;
32                 }
33             }
34             if (check == 0) break;
35         }
36         cout << ans.substr(0, ans.length() - 1) << endl;
37     }
38     return 0;
39 }

```

## ADV-15. 最大乘积

### 问题描述

对于n个数，从中取出m个数，如何取使得这m个数的乘积最大呢？

### 输入格式

第一行一个数表示数据组数

每组输入数据共2行：

第1行给出总共的数字的个数n和要取的数的个数m， $1 \leq n \leq m \leq 15$ ，

第2行依次给出这n个数，其中每个数字的范围满足： $a[i]$ 的绝对值小于等于4。

输出格式

每组数据输出1行，为最大的乘积。

样例输入

1

5 5

1 2 3 4 2

样例输出

48

分析：负数要想选择，必须两个两个的选择。正数只需要一个一个选择就好~所以把数组按照从小到大排序，这样负数就在最左边，正数就在最右边啦~当还可以选择两个或者两个以上数字的时候~比较两个左边的数字相乘会不会比右边的数字相乘的结果大~如果大的话，那就选左边那两个负数~如果只能选择一个数字了，或者左边两个数的乘积不比右边两个数的乘积大~那么就选择最右边那个最大数字就好~相乘得到结果~~

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int main() {
5      int cnt;
6      cin >> cnt;
7      for(int i = 0; i < cnt; i++) {
8          int n, m;
9          cin >> n >> m;
10         int *a = new int [n];
11         int ans = 1;
12         for(int j = 0; j < n; j++) {
13             cin >> a[j];
14         }
15         sort(a, a+n);
16         int p = 0, q = n - 1;
17         while(p <= n-1 && q >= 0 && m > 0) {
18             if((a[p] * a[p+1] > a[q] * a[q-1]) && m >= 2) {
19                 ans = ans * (a[p] * a[p+1]);
20                 p += 2;
21                 m -= 2;
22             } else {
23                 ans = ans * (a[q]);
24                 q--;
25                 m--;
```

```

26         }
27     }
28     cout << ans << endl;
29 }
30 return 0;
31 }

```

## ADV-16. 和最大子序列

### 问题描述

对于一个给定的长度为N的整数序列A，它的“子序列”的定义是：A中非空的一段连续的元素（整数）。你要完成的任务是，在所有可能的子序列中，找到一个子序列，该子序列中所有元素的和是最大的（跟其他所有子序列相比）。程序要求你输出这个最大值。

### 输入格式

输入文件的第一行包含一个整数N，第二行包含N个整数，表示A。

其中

$1 \leq N \leq 100000$

$-10000 \leq A[i] \leq 10000$

### 输出格式

输出仅包含一个整数，表示你算出的答案。

### 样例输入

```

5
3 -2 3 -5 4

```

### 样例输入

```

4

```

分析：sum为某一段和，sum=0 从0号开始向前，遇到一个数字，若该数字加起来不会让sum变成负数，则加上它，若加起来变成负数的话，就丢掉前面所有的数字，置为0，用maxn记录sum出现过最大的数字即可～

```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int n, a[100005] = {0}, sum = 0, maxn = 0;
6      scanf("%d", &n);
7      for (int i = 0; i <= n; i++) {
8          scanf("%d", &a[i]);
9          if (sum + a[i] >= 0) sum = sum + a[i];
10         else sum = 0;
11         maxn = max(maxn, sum);
12     }
13     cout << maxn;
14     return 0;

```



## 问题描述

输入的实数满足如下要求: (1) 小数点前的整数部分最多100位, (2) 小数点后的小数部分最多100位.

两行字符串, 每行都是一个合法的实数。合法的意思是指: 整数部分的值如果大于零, 则最高位数字必定大于零。如果整数部分的值为零, 则整数部分只有一个零。小数部分尾部可以有任意多的零。可以没有小数部分, 此时也没有小数点。如果有小数点, 则至少需要有一位小数部分, 且允许是零。

相加结果。注意: 小数部分末尾如果有连续的0, 则它们都是有效数字, 不能舍去。如果是两个整数相加, 则结果仍为整数而没有小数部分。

[illegible]

8.999999999999999999999999999

[illegible]

10.000999999999999999999999999

分析：1.高精度模拟竖式加法，整数部分相加，小数部分相加  
2.整数部分，位数不够，前面补0；小数部分位数不够，后面补0  
3.小数部分如果有进位，要截下来加到整数上去~

1

```

2  #include <string>
3  using namespace std;
4  string add1(string s1, string s2) {
5      int len1 = s1.length(), len2 = s2.length();
6      if (len1 < len2) {
7          string t(len2 - len1, '0');
8          s1 = t + s1;
9      } else if (len2 < len1) {
10         string t(len1 - len2, '0');
11         s2 = t + s2;
12     }
13     string ans = s1;
14     int car = 0;
15     for (int i = s1.length() - 1; i >= 0; i--) {
16         ans[i] = (s1[i] - '0' + s2[i] - '0' + car) % 10 + '0';
17         car = (s1[i] - '0' + s2[i] - '0' + car) / 10;
18     }
19     if (car) ans = (char) (car + '0') + ans;
20     return ans;
21 }
22 string add2(string s1, string s2) {
23     int len1 = s1.length(), len2 = s2.length();
24     if (len1 < len2) {
25         string t(len2 - len1, '0');
26         s1 = s1 + t;
27     } else if (len2 < len1) {
28         string t(len1 - len2, '0');
29         s2 = s2 + t;
30     }
31     string ans = s1;
32     int car = 0;
33     for (int i = s1.length() - 1; i >= 0; i--) {
34         ans[i] = (s1[i] - '0' + s2[i] - '0' + car) % 10 + '0';
35         car = (s1[i] - '0' + s2[i] - '0' + car) / 10;
36     }
37     if (car) ans = (char) (car + '0') + ans;
38     return ans;
39 }
40 int main() {
41     string a, b;
42     while (cin >> a >> b) {
43         string a1, a2, b1, b2;
44         int i, j;
45         for (i = 0; i < a.length() && a[i] != '.'; i++);
46         for (j = 0; j < b.length() && b[j] != '.'; j++);
47         a1 = a.substr(0, i);
48         b1 = b.substr(0, j);
49
50         if (i == a.length()) a2 = "";

```

```

51         else a2 = a.substr(i + 1, a.length() - i - 1);
52         if (j == b.length()) b2 = "";
53         else b2 = b.substr(j + 1, b.length() - j - 1);
54
55         string ans1 = add1(a1, b1);
56         string ans2 = add2(a2, b2);
57
58         if (ans2.length() > max(a2.length(), b2.length())) {
59             ans2 = ans2.substr(1, ans2.length() - 1);
60             ans1 = add1(ans1, "1");
61         }
62         if (ans2.length() > 0) ans2 = "." + ans2;
63         cout << ans1 << ans2 << endl;
64     }
65     return 0;
66 }

```

## ADV-20. 交换Easy

### 问题描述

给定N个整数组成的序列，每次交换当前第x个与第y个整数，要求输出最终的序列。

### 输入格式

第一行为序列的大小N( $1 \leq N \leq 1000$ )和操作个数M( $1 \leq M \leq 1000$ )。

第二行包含N个数字，表示初始序列。

接下来M行，每行两个整数x,y ( $1 \leq x, y \leq N$ )，表示要交换的两个整数。在一次交换中，如果x和y相等，则不会改变序列的内容。

### 输出格式

输出N行，为交换后的序列中的数。

### 样例输入

```

5 2
1 2 3 4 5
1 2
3 4

```

### 样例输出

```

2
1
4
3

```

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, m;
5      cin >> n >> m;
6      int *a = new int[n];
7      for(int i = 0; i < n; i++)
8          cin >> a[i];
9      for(int i = 0; i < m; i++) {
10         int p, q;
11         cin >> p >> q;
12         swap(a[p-1], a[q-1]);
13     }
14     for(int i = 0; i < n; i++)
15         cout << a[i] << endl;
16     delete [] a;
17     return 0;
18 }

```

## ADV-21. 多项式输出

### 问题描述

一元 $n$ 次多项式可用如下的表达式表示：

$$f(x)=a[n]x^n+a[n-1]x^{(n-1)}+...+a[1]x+a[0], a[n] \neq 0$$

其中， $a[i]x^i$ 称为 $i$ 次项， $a[i]$ 称为 $i$ 次项的系数。给出一个一元多项式各项的次数和系数，请按照如下规定的格式要求输出该多项式：

1. 多项式中自变量为 $x$ ，从左到右按照次数递减顺序给出多项式。
2. 多项式中只包含系数不为0的项。
3. 如果多项式 $n$ 次项系数为正，则多项式开头不出现“+”号，如果多项式 $n$ 次项系数为负，则多项式以“-”号开头。
4. 对于不是最高次的项，以“+”号或者“-”号连接此项与前一项，分别表示此项系数为正或者系数为负。紧跟一个正整数，表示此项系数的绝对值（如果一个高于0次的项，其系数的绝对值为1，则无需输出1）。如果 $x$ 的指数大于1，则接下来紧跟的指数部分的形式为“ $x^b$ ”，其中 $b$ 为 $x$ 的指数；如果 $x$ 的指数为1，则接下来紧跟的指数部分形式为“ $x$ ”；如果 $x$ 的指数为0，则仅需输出系数即可。
5. 多项式中，多项式的开头、结尾不含多余的空格。

### 输入格式

输入共有2行

第一行1个整数， $n$ ，表示一元多项式的次数。

第二行有 $n+1$ 个整数，其中第 $i$ 个整数表示第 $n-i+1$ 次项的系数，每两个整数之间用空格隔开。

$1 \leq n \leq 100$ ，多项式各次项系数的绝对值均不超过100。

### 输出格式

输出共1 行，按题目所述格式输出多项式。

样例输入

5  
100 -1 1 -3 0 10

样例输出

$100x^5 - x^4 + x^3 - 3x^2 + 10$

样例输入

3  
-50 0 0 1

样例输出

$-50x^3 + 1$

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, a[1000];
5      cin >> n;
6      for (int i = n; i >= 0; i--) {
7          cin >> a[i];
8          if (a[i] != 0) {
9              if (i != 0) {
10                 if (i != n && a[i] > 0) cout << "+";
11                 if (a[i] == -1) cout << "-";
12                 else if (a[i] != 1) cout << a[i];
13                 cout << "x";
14                 if (i != 1) cout << "^" << i;
15             } else {
16                 if (a[i] > 0) cout << "+";
17                 cout << a[i];
18             }
19         }
20     }
21     return 0;
22 }
```

## ADV-61. 矩阵乘方

问题描述

给定一个矩阵A,一个非负整数b和一个正整数m, 求A的b次方除m的余数。

其中一个 $n \times n$ 的矩阵除m的余数得到的仍是一个 $n \times n$ 的矩阵, 这个矩阵的每一个元素是原矩阵对应位置上的数除m的余数。

要计算这个问题, 可以将A连乘b次, 每次都对m求余, 但这种方法特别慢, 当b较大时无法使用。下面给出一种较快的算法(用 $A^b$ 表示A的b次方):

若 $b=0$ ，则 $A^b \% m = I \% m$ 。其中 $I$ 表示单位矩阵。

若 $b$ 为偶数，则 $A^b \% m = (A^{b/2} \% m)^2 \% m$ ，即先把 $A$ 乘 $b/2$ 次方对 $m$ 求余，然后再平方后对 $m$ 求余。

若 $b$ 为奇数，则 $A^b \% m = (A^{b-1} \% m) * a \% m$ ，即先求 $A$ 乘 $b-1$ 次方对 $m$ 求余，然后再乘 $A$ 后对 $m$ 求余。

这种方法速度较快，请使用这种方法计算 $A^b \% m$ ，其中 $A$ 是一个 $2 \times 2$ 的矩阵， $m$ 不大于10000。

输入格式

输入第一行包含两个整数 $b, m$ ，第二行和第三行每行两个整数，为矩阵 $A$ 。

输出格式

输出两行，每行两个整数，表示 $A^b \% m$ 的值。

样例输入

2 2

1 1

0 1

样例输出

1 0

0 1

分析：1.用快速幂的解法递归下去即可

2.按照题目测试数据，矩阵的0次方，应该为全部数字为0，矩阵的1次方，矩阵不变~

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int m;
5  vector<int> mul(vector<int> a, vector<int> b) {
6      vector<int> ans(5);
7      ans[1] = (a[1] * b[1] + a[2] * b[3]) % m;
8      ans[2] = (a[1] * b[2] + a[2] * b[4]) % m;
9      ans[3] = (a[3] * b[1] + a[4] * b[3]) % m;
10     ans[4] = (a[3] * b[2] + a[4] * b[4]) % m;
11     return ans;
12 }
13 vector<int> f(vector<int> v, int b) {
14     vector<int> minn(5), nulln(5);
15     minn[1] = minn[4] = 1;
16     if (b == 0) {
17         return mul(nulln, minn);
18     } else if (b == 1) {
19         return mul(v, minn);
20     } else if (b % 2 == 0) {
21         vector<int> t(5);
22         t = f(v, b / 2);
23         return mul(t, t);
24     } else {
25         return mul(f(v, b - 1), v);
26     }
```

```

26     }
27 }
28 int main() {
29     vector<int> v(5), ans;
30     int b;
31     cin >> b >> m;
32     cin >> v[1] >> v[2] >> v[3] >> v[4];
33     ans = f(v, b);
34     printf("%d %d\n%d %d\n", ans[1], ans[2], ans[3], ans[4]);
35     return 0;
36 }

```

## ADV-62. 夺宝奇兵(动态规划)

### [题目描述]

在一座山上,有很多很多珠宝,它们散落在山底通往山顶的每条道路上,不同道路上的珠宝的数目也各不相同.下图为一张藏宝地图:

```

7
3 8
8 1 0
2 7 4 4
4 5 2 6 5

```

“夺宝奇兵”从山下出发,到达山顶,如何选路才能得到最多的珠宝呢?在上图所示例子中,按照5->7->8->3->7的顺序,将得到最大值30

### [输入]

第一行正整数N( $100 \geq N > 1$ ),表示山的高度

接下来有N行非负整数,第i行有i个整数( $1 \leq i \leq N$ ),表示山的第i层上从左到右每条路上的珠宝数目

### [输出]

一个整数,表示从山底到山顶的所能得到的珠宝的最大数目.

### [样例输入]

```

5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5

```

### [样例输出]

分析：经典的动态规划之数字三角形问题~~将输入存储入二维数组后，从上到下遍历，对于最左边的一列，等于 $a[i][j] += a[i-1][j]$ ；对于最右边的一列， $a[i][j] += a[i-1][j-1]$ ；对于其他的中间列，等于 $a[i][j] += \max(a[i-1][j-1], a[i-1][j])$ ；最后看最后一行的最大值是多少，即可找到这样一条最大数目珠宝的路径~

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int a[101][101];
7      for(int i = 0; i < n; i++) {
8          for(int j = 0; j <= i; j++) {
9              cin >> a[i][j];
10         }
11     }
12     int ans = 0;
13     for(int i = 1; i < n; i++) {
14         for(int j = 0; j <= i; j++) {
15             if(j == 0)
16                 a[i][j] += a[i-1][j];
17             else if(j == i)
18                 a[i][j] += a[i-1][j-1];
19             else
20                 a[i][j] += max(a[i-1][j-1], a[i-1][j]);
21         }
22     }
23     for(int i = 1; i < n; i++) {
24         ans = ans > a[n-1][i] ? ans : a[n-1][i];
25     }
26     cout << ans;
27     return 0;
28 }

```

## ADV-63. 利息计算

### 问题描述

编制程序完成下述任务：接受两个数，一个为用户一年期定期存款金额，一个为按照百分比格式表示的利率；程序计算一年期满后本金与利息总额。说明：（1）存款金额以人民币元为单位，可能精确到分；（2）输入利率时不需要输入百分号，例如一年期定期存款年利率为2.52%，用户输入2.52即可；（3）按照国家法律，存款利息所得需缴纳20%的所得税，计算结果时所得税部分应扣除。

### 输入格式

输入一行，包含两个实数，分别表示本金和年利率。



输出格式

输出一行，包含一个实数，保留到小数点后两位，表示一年后的本金与利息和。

样例输入

10000 2.52

样例输出

10201.60

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      double m, r;
5      cin >> m >> r;
6      printf("%.2f", m + 0.8 * 0.01 * m * r);
7      return 0;
8  }
```

## ADV-65. 格子位置

问题描述

输入三个自然数 $N, i, j$  ( $1 \leq i \leq N, 1 \leq j \leq N$ )，输出在一个 $N \times N$ 格的棋盘上，与格子 $(i, j)$ 同行、同列、同一对角线的所有格子的位置。

输入格式

输入共三行，分别输入自然数 $N, i, j$ 。其中保证 $N \leq 24$ 且 $1 \leq i \leq N, 1 \leq j \leq N$ 。

输出格式

输出共四行。第一行为与格子 $(i, j)$ 同行的所有格子的位置，第二行为与格子 $(i, j)$ 同列的所有格子的位置，第三行为从左上到右下对角线上的格子的位置，第四行为从左下到右上对角线上的格子的位置。

样例输入

4  
2  
3

样例输出

(2,1) (2,2) (2,3) (2,4)  
(1,3) (2,3) (3,3) (4,3)  
(1,2) (2,3) (3,4)  
(4,1) (3,2) (2,3) (1,4)

输入输出样例解释

$n=4, i=2, j=3$ 表示了棋盘中的第二行第三列的格子，如下图：

第1列	第2列	第3列	第4列	
				第1行
		(2,3)		第2行
				第3行
				第4行

(2,1) (2,2) (2,3) (2,4) {同一行上格子的位置}  
 (1,3) (2,3) (3,3) (4,3) {同列上格子的位置}  
 (1,2) (2,3) (3,4) {左上到右下对角线上的格子位置}  
 (4,1) (3,2) (2,3) (1,4) {左下到右上对角线上的格子位置}

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, x, y, i, j;
5      cin >> n >> x >> y;
6      for (i = 1; i <= n; i++)
7          printf("(%d,%d)", x, i);
8      printf("\n");
9      for (i = 1; i <= n; i++)
10         printf("(%d,%d)", i, y);
11     printf("\n");
12     for (int i = 1; i <= n; i++) {
13         for (int j = 1; j <= n; j++) {
14             if (i - x == j - y)
15                 printf("(%d,%d)", i, j);
16         }
17     }
18     printf("\n");
19     for (int i = n; i >= 1; i--) {
20         for (int j = 1; j <= n; j++) {
21             if (i - x == -1 * (j - y))
22                 printf("(%d,%d)", i, j);
23         }
24     }
25     return 0;
26 }
```

## ADV-66. 阮小二买彩票

### 问题描述

在同学们的帮助下，阮小二是变的越来越懒了，连算账都不愿意自己亲自动手了，每天的工作就是坐在电脑前看自己的银行账户的钱是否有变多。可是一段时期观察下来，阮小二发现自己账户的钱增长好慢啊，碰到节假日的时候连个铜板都没进，更郁闷的是这些天分文不进就算了，可恨的是银行这几天还有可能“落井下石”(代扣个人所得税)，看着自己账户的钱被负增长了，阮小二就有被割肉的感觉(太痛苦了！)，这时阮小二最大的愿望无疑是以最快的速度日进斗金，可什么方法能够日进斗金

呢？抢银行(老本行)？不行，太危险，怕有命抢没命花；维持现状？受不了，捞钱太慢了！想来想去，抓破脑袋之后，终于想到了能快速发家致富的法宝——买彩票，不但挣了钱有命花，运气好的话，可以每天中他个几百万的，岂不爽哉！抱着这种想法，阮小二开始了他的买彩票之旅。想法是“好的”（太天真了OR太蠢了），可是又发现自己的数学功底太差，因为不知道数字都有哪些组合排列？那现在就请同学们写个递归程序，帮助阮小二解决一下这个问题吧！

输入格式

不超过6位数的正整数N，注意：构成正整数N的数字可重复

输出格式

组成正整数N的所有位数的全排列，这些排列按升序输出，每个排列占一行。

注意：输出数据中不能有重复的排列

样例输入

123

样例输出

123

132

213

231

312

321

样例输入

3121

样例输出

1123

1132

1213

1231

1312

1321

2113

2131

2311

3112

3121

3211

样例输入

4003

样例输出

0034  
0043  
0304  
0340  
0403  
0430  
3004  
3040  
3400  
4003  
4030  
4300

分析：用next\_permutation(s.begin(), s.end())输出s字符串的全排列~~在头文件<algorithm>里面  
~~~//要不是你因为好用，谁愿意记住辣么长的单词- -||

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int main() {
5      string s;
6      cin >> s;
7      int len = s.length();
8      sort(s.begin(), s.end());
9      do {
10         cout << s << endl;
11     }while(next_permutation(s.begin(), s.end()));
12     return 0;
13 }
```

## ADV-68. 企业奖金发放

企业发放的奖金根据利润提成。利润低于或等于10万元时，奖金可提10%；利润高于10万元，低于20万元时，

低于10万元的部分按10%提成，高于10万元的部分，可提成7.5%；20万到40万之间时，高于20万元的部分，可提成5%；

40万元到60万元之间时高于40万元的部分，可提成3%；60万到100万之间时，高于60万元的部分，可提成1.5%；

高于100万元时，超过100万元的部分按1%提成。从键盘输入当月利润，求应发放奖金总数？

（保留两位小数）利润的大小在double以内

样例输入

210000

样例输出

18000.00

```

1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main() {
5      double a;
6      cin >> a;
7      double b = 0;
8      int s[6] = {1000000, 600000, 400000, 200000, 100000, 0};
9      double t[6] = {0.01, 0.015, 0.03, 0.05, 0.075, 0.1};
10     for(int i = 0; i < 6; i++) {
11         if(a - s[i] >= 0) {
12             b += (a - s[i]) * t[i];
13             a = s[i];
14         }
15     }
16     printf("%.2f", b);
17     return 0;
18 }

```

## ADV-69. 质因数

将一个正整数N( $1 < N < 32768$ )分解质因数。例如，输入90，打印出 $90=2*3*3*5$ 。

样例输入

66

样例输出

66=2\*3\*11

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      for(int i = 2; i <= n; i++) {
7          if(i != n && n % i == 0) {
8              cout << n << "=" << i;
9          } else if(i == n) {
10             cout << n << "=" << 1 << "*" << i;
11             return 0;
12         }
13     }
14     while(n != 1) {
15         for(int i = 2; i <= n; i++) {
16             if(n % i == 0) {
17                 cout << "*" << i;
18                 n = n / i;
19                 break;

```

```

20         }
21     }
22 }
23 return 0;
24 }

```

## ADV-70. 冒泡法排序

输入10个数，用“冒泡法”对10个数排序（由小到大）这10个数字在100以内。

样例输入

1 3 6 8 2 7 9 0 4 5

样例输出

0 1 2 3 4 5 6 7 8 9

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a[10];
5      for(int i = 0; i < 10; i++) {
6          cin >> a[i];
7      }
8      for(int i = 0; i < 9; i++) {
9          for(int j = 0; j < 9 - i; j++) {
10             if(a[j] > a[j+1]) {
11                 swap(a[j], a[j+1]);
12             }
13         }
14     }
15     for(int i = 0; i < 10; i++) {
16         cout << a[i] << " ";
17     }
18     return 0;
19 }

```

## ADV-71. 判断回文

编程判断一个字符串是否是回文，当字符串是回文时，输出字符串：yes!，否则输出字符串：no!。所谓回文即正向与反向的拼写都一样，如adgda。 长度在100以内，且全为小写字母

样例输入

adgda

样例输出

yes!

```

1  #include <iostream>

```

```

2   using namespace std;
3   int main() {
4       string s;
5       cin >> s;
6       int i = 0, j = s.length()-1;
7       while(i < j) {
8           if(s[i] != s[j]) {
9               cout << "no!";
10              return 0;
11          }
12          i++;
13          j--;
14      }
15      cout << "yes!";
16      return 0;
17  }

```

## ADV-72. 一元一次方程

输入一元一次方法的 $ax+b=0$ 的解。且数据均在double类型以内,且一定有解（保留2位小数）

样例输入

2 6

样例输出

-3.00

```

1   #include <iostream>
2   using namespace std;
3   int main() {
4       double ans;
5       double a, b;
6       cin >> a >> b;
7       ans = (0 - b) / a;
8       printf("%.2lf", ans);
9       return 0;
10  }

```

## ADV-73. 数组输出

输入一个3行4列的数组，找出该数组中绝对值最大的元素、输出该元素及其两个下标值。如有多个输出行号最小的，还有多个的话输出列号最小的。

样例输入

1 2 3 5

-2 5 8 9

6 -7 5 3

样例输出

9 2 4

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int m = 0, x = 1, y = 1;
5      for(int i = 0; i < 3; i++) {
6          for(int j = 0; j < 4; j++) {
7              int temp;
8              cin >> temp;
9              if(temp < 0)
10                 temp = 0 - temp;
11             if(temp > m) {
12                 m = temp;
13                 x = i+1;
14                 y = j+1;
15             }
16         }
17     }
18     cout << m << " " << x << " " << y;
19     return 0;
20 }
```

## ADV-74. 计算整数因子

问题描述

输入一个整数，输出其所有质因子。

输入格式

输入只有一行，包含一个整数n。

输出格式

输出一行，包含若干个整数，为n的所有质因子，按照从小到大的顺序排列。

样例输入

6

样例输出

2 3

数据规模和约定

$1 \leq n \leq 10000$ 。

```
1  #include <iostream>
2  using namespace std;
```



```

3  int main() {
4      int n;
5      cin >> n;
6      for(int i = 2; i < n; i++) {
7          int flag = 0;
8          for(int j = 2; j * j <= i; j++) {
9              if(i % j == 0)
10                 flag = 1;
11            }
12            if(flag == 0 && n % i == 0)
13                cout << i << " ";
14        }
15        return 0;
16    }

```

## ADV-75. 简单计算器

### 问题描述

编程模拟计算器的加、减、乘、除功能，根据用户输入的运算符，对两个数进行运算。(要求switch语句)

### 输入格式

输入只有一行，用空格隔开的运算符和两个运算数，运算符一定是+, -, \*, /之一，运算数一定是绝对值不超过200的整数，当运算符为除号时，除数不为0并第一个数一定是第二个数的整数倍。

### 输出格式

输出只有一行，包含一个整数，表示运算结果。

### 样例输入

/ 6 2

### 样例输出

3

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      char c;
5      int a, b;
6      cin >> c >> a >> b;
7      switch(c) {
8          case '+':
9              cout << a + b;
10             break;
11          case '-':
12              cout << a - b;
13             break;

```

```

14         case '*':
15             cout << a * b;
16             break;
17         case '/':
18             cout << a / b;
19             break;
20     }
21     return 0;
22 }

```

## ADV-76. GDP计算

### 问题描述

设我国国民生产总值的年增产率为10%，计算n年后我国国民生产总值与现在的比是多少。计算公式为： $P=(1+r)^n$ ,  $r$ 为年增产率， $n$ 为年数， $P$ 为n年后国民生产总值与现在相比的倍数。

### 输入格式

输入一个数 $n$ ( $1 \leq n \leq 300$ )。

### 输出格式

输出一个数 $P$ ，保留2位小数。

### 样例输入

10

### 样例输出

2.59

```

1  #include <iostream>
2  #include <cmath>
3  #include <cstdio>
4  using namespace std;
5  int main() {
6      int n;
7      cin >> n;
8      double ans = pow(1.1, n);
9      printf("%.2lf", ans);
10     return 0;
11 }

```

## ADV-77. 统计平均成绩

有4个学生，上4门课，要求输入全部学生的各门课成绩，并分别求出每门课的平均成绩。(保留2位小数)

括号里是解释内容，不用输入输出。输入的所有数都为0到100之间（包括端点）的整数

### 样例输入

(输入第1个学生的4门课成绩) 94 78 87 96

(输入第2个学生的4门课成绩) 66 87 75 69

(输入第3个学生的4门课成绩) 100 98 89 77

(输入第4个学生的4门课成绩) 82 73 67 54

样例输出

(第1门课的平均成绩是) 85.50

(第2门课的平均成绩是) 84.00

(第3门课的平均成绩是) 79.50

(第4门课的平均成绩是) 74.00

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main() {
5      double c1 = 0.0, c2 = 0.0, c3 = 0.0, c4 = 0.0;
6      double t1, t2, t3, t4;
7      for(int i = 0; i < 4; i++) {
8          cin >> t1 >> t2 >> t3 >> t4;
9          c1 += t1;
10         c2 += t2;
11         c3 += t3;
12         c4 += t4;
13     }
14     printf("%.2lf\n%.2lf\n%.2lf\n%.2lf", c1 / 4, c2 / 4, c3 / 4, c4 /
15     4);
16     return 0;
17 }
```

## ADV-78. 最长单词

编写一个函数，输入一行字符，将此字符串中最长的单词输出。

输入仅一行，多个单词，每个单词间用一个空格隔开。单词仅由小写字母组成。所有单词的长度和不超过100000。如有多个最长单词，输出最先出现的。

样例输入

I am a student

样例输出

student

```
1  #include <iostream>
2  using namespace std;
3  int main() {
```

```

4     string s;
5     string temp;
6     int mmax = 0;
7     while(cin >> temp) {
8         int len = temp.length();
9         if(mmax < len) {
10            mmax = len;
11            s = temp;
12        }
13    }
14    cout << s;
15    return 0;
16 }

```

## ADV-79. 时间转换

输入n分钟换算成天、小时和分输出。例如4880分钟，可换算成3天9小时20分。

样例输入

4880

样例输出

3 9 20

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      cout << n / 1440 << " ";
7      n = n % 1440;
8      cout << n / 60 << " ";
9      n = n % 60;
10     cout << n;
11     return 0;
12 }

```

## ADV-80. 选最大数

输入3个整数a、b、c，（数的范围是[1,10000]）输出其中最大的数。（用指针实现）

样例输入

2 5 1

样例输出

5

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b, c;
5      cin >> a >> b >> c;
6      int *p = a;
7      if(*p < b)
8          p = &b;
9      if(*p < c)
10         p = &c;
11     return 0;
12 }

```

## ADV-81. 数的运算

输入一个正整数（范围[1..10000]），打印其平方（不保留小数位）、平方根、倒数。(用指针实现，保留2位小数，输出每个数之间以一个空格隔开)

样例输入

2

样例输出

4 1.41 0.50

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cmath>
4  using namespace std;
5  void func1(int *p) {
6      cout << (*p)*(*p) << " ";
7  }
8
9  void func2(int *p) {
10     printf("%.2lf ", sqrt(*p));
11 }
12
13 void func3(int *p) {
14     printf("%.2lf", 1.0 / (*p));
15 }
16
17 int main() {
18     int n;
19     cin >> n;
20     int *p = &n;
21     func1(p);
22     func2(p);
23     func3(p);
24     return 0;

```

## ADV-82. 填充蛋糕

编程计算涂满高为2，半径为r的圆形蛋糕表面，需要多少表面积的奶油(只要涂上表面和侧面)

读入一个数r，输出需要奶油的表面积，结果保留一位小数

样例输入

5.0

样例输出

141.4

```

1  #include <iostream>
2  #include <cmath>
3  #include <cstdio>
4  using namespace std;
5  int main() {
6      double r;
7      cin >> r;
8      double ans = r * M_PI * r + 4 * M_PI * r;
9      printf("%.1lf", ans);
10     return 0;
11 }
12
13 或者用atan(1)*4代替:
14
15 #include <iostream>
16 #include <cmath>
17 #include <cstdio>
18 using namespace std;
19 int main() {
20     double r;
21     cin >> r;
22     double ans = r * atan(1) * 4 * r + 16 * atan(1) * r;
23     printf("%.1lf", ans);
24     return 0;
25 }
```

## ADV-83. 寻找三位数

问题描述

将1, 2, ..., 9共9个数分成三组，分别组成三个三位数，且使这三个三位数构成

1: 2: 3的比例，试求出所有满足条件的三个三位数。

例如：三个三位数192, 384, 576满足以上条件。

输入格式

无输入文件

输出格式

输出每行有三个数，为满足题设三位数。各行为满足要求的不同解。

分析：先确定第一个数字，然后判断这个数字的两倍数和三倍数是否满足条件~用book数组标记当前数字是否已经出现过~

```
1  #include <iostream>
2  using namespace std;
3  bool judge(int i, int j, int k) {
4      int book[10] = {0};
5      book[0] = 1;
6      if(i == j || j == k || i == k)
7          return false;
8      book[i] = 1, book[j] = 1, book[k] = 1;
9      int one = i * 100 + j * 10 + k;
10     int two = 2 * one;
11     int three = 3 * one;
12     if(two >= 666 || three >= 987)
13         return false;
14     if(book[two/100] == 1)
15         return false;
16     else
17         book[two/100] = 1;
18
19     if(book[two%100/10] == 1)
20         return false;
21     else
22         book[two%100/10] = 1;
23
24     if(book[two%10] == 1)
25         return false;
26     else
27         book[two%10] = 1;
28
29     if(book[three/100] == 1)
30         return false;
31     else
32         book[three/100] = 1;
33
34     if(book[three%100/10] == 1)
35         return false;
36     else
37         book[three%100/10] = 1;
38
39     if(book[three%10] == 1)
```

```

40         return false;
41     else
42         return true;
43 }
44 int main() {
45     for(int i = 1; i <= 3; i++) {
46         for(int j = 1; j <= 9; j++) {
47             for(int k = 1; k <= 9; k++) {
48                 if(judge(i, j, k) == true) {
49                     int one = i * 100 + j * 10 + k;
50                     cout << one << " " << one * 2 << " " << one * 3 <<
endl;
51                 }
52             }
53         }
54     }
55     return 0;
56 }

```

## ADV-84. 图形输出

编写一程序，在屏幕上输出如下内容：

```

X | X | X
--+--+
  |  |
--+--+
O | O | O

```

注意：本题请同学们严格按照图形的格式输出，对齐，其中X和O为大写，否则系统会判为错误。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      cout << " X | X | X" << endl << "---+---+---" << endl << "   |   |
" << endl << "---+---+---" << endl << " O | O | O" << endl;
5      return 0;
6  }

```

## ADV-85. 算术运算

编写一程序，接受用户输入的两个整数，并计算它们的和、差、

积、商，程序运行时候输入输出例子如下所示。

样例输入：

3 5



样例输出：

3+5=8

3-5=-2

3\*5=15

3/5=0

注意：输出要严格按照+-\* /的顺序，分四行输出，而且中间不能有空格，否则系统会判为错误。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b;
5      cin >> a >> b;
6      cout << a << "+" << b << "=" << a+b << endl;
7      cout << a << "-" << b << "=" << a-b << endl;
8      cout << a << "*" << b << "=" << a*b << endl;
9      cout << a << "/" << b << "=" << a/b << endl;
10     return 0;
11 }
```

## ADV-86. 格式化数据输出

编制程序，输出下述数据。说明：（1）表中数据来自总参谋部测绘局编制的

《世界地图集》（星球地图出版社，2004年1月第2版），数据可能已不准确；

（2）面积单位为万平方公里，人口单位为万人，GDP单位为十亿美元；

（3）表中所有数据都必须以变量的形式保存；（4）如果不知道每字段宽度到底为多少，请仔细数数作为分隔标记的短横数目。

-----  
COUNTRY AREA(10K km2) POP.(10K) GDP(Billion\$)

-----  
China 960.00 129500.00 1080.00

Iceland 10.30 27.57 8.20

India 297.47 97000.00 264.80

Madagascar 62.70 1635.00 3.60

Maldiva 0.0298 27.80 0.23  
-----

注意：输出时空格与短线的数量要与上面格式严格一致，否则系统会判为错误。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      cout << "-----\nCOUNTRY
      AREA(10K km2)  POP.(10K)  GDP(Billion$)\n-----
      -----\nChina                960.00  129500.00
1080.00\n";
5      cout << "Iceland                10.30    27.57  8.20\nIndia
      297.47   97000.00  264.80\nMadagascar        62.70    1635.00
3.60\nMaldiver                0.0298    27.80  0.23\n";
6      cout << "-----\n";
7      return 0;
8  }

```

## ADV-87. 利息计算

编制程序完成下述任务：接受两个数，一个为用户一年期定期存款金额，一个为按照百分比格式表示的利率；程序计算一年期满

后本金与利息总额。说明：（1）存款金额以人民币元为单位，可能精确到分；（2）输入利率时不需要输入百分号，例如一年期定期存款年利率

为2.52%，用户输入2.52即可；（3）按照国家法律，存款利息所得需缴纳20% 的所得税，计算结果时所得税部分应扣除。要求输出小数点后严格

保留两位小数。

样例输入

10000 2.52

样例输出

10201.60

```

1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main() {
5      double a, p;
6      cin >> a >> p;
7      double ans = a + a * p * 0.01 * 0.8;
8      printf("%.2f", ans);
9      return 0;
10 }

```

## ADV-88. 输出正反三角形

使用循环结构打印下述图形，打印行数n由用户输入。图中每行事实上包括两部分，中间间隔空格字符数m也由用户输入。

样例输入n,m:

样例输入n,m:

5 4

样例输出:

```
* *****
*** *****
***** *****
***** ***
***** * *
```

注意：两行之间没有空行。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, m;
5      cin >> n >> m;
6      for(int i = 1; i <= n; i++) {
7          for(int j = 1; j <= n - i; j++)
8              cout << " ";
9          for(int j = 1; j <= 2 * i - 1; j++)
10             cout << "*";
11         for(int j = 1; j <= m; j++)
12             cout << " ";
13         for(int j = 1; j <= 2 * (n - i + 1) - 1; j++)
14             cout << "*";
15         cout << endl;
16     }
17     return 0;
18 }
```

## ADV-89. 输出九九乘法表

编制程序，按照下述格式打印九九乘法表。

输出样例：

Nine-by-nine Multiplication Table

-----

1 2 3 4 5 6 7 8 9

-----

1 1

2 2 4  
3 3 6 9  
4 4 8 12 16  
5 5 10 15 20 25  
6 6 12 18 24 30 36  
7 7 14 21 28 35 42 49  
8 8 16 24 32 40 48 56 64  
9 9 18 27 36 45 54 63 72 81

---

注意：表头的大小写要和样例一致，短线“-”个数要与样例中一致，否则系统会判为错误。

注意：

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      cout << "Nine-by-nine Multiplication Table" << endl;
5      cout << "-----" << endl;
6      cout << "    1    2    3    4    5    6    7    8    9" << endl;
7      cout << "-----" << endl;
8      cout << " 1    1\n2    2    4\n3    3    6    9\n4    4    8    12    16\n5    5
10 15    20    25\n6    6    12    18    24    30    36\n";
9      cout << "7    7    14    21    28    35    42    49\n8    8    16    24    32    40    48
56 64\n9    9    18    27    36    45    54    63    72    81\n";
10     cout << "-----";
11     return 0;
12 }
```

## ADV-90. 输出日历

按照下述格式打印2006年12月日历：

Calendar 2006-12

-----

Su Mo Tu We Th Fr Sa

-----

1 2  
3 4 5 6 7 8 9  
10 11 12 13 14 15 16  
17 18 19 20 21 22 23

24 25 26 27 28 29 30

31

注意：表头和表中的英文字符大小写要与题目中一致，短线“-”个数要与题目中一致，否则系统会判为错误。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      cout << "Calendar 2006-12\n-----\nSu  Mo  Tu  We
Th  Fr  Sa\n-----\n";
5      cout << "
1 2\n3 4 5 6 7 8 9\n10
11 12 13 14 15 16\n17 18 19 20 21 22 23\n";
6      cout << "24 25 26 27 28 29 30\n31\n-----";
7      return 0;
8  }
```

## ADV-91. 素数判断

编写一函数IsPrime，判断某个大于2的正整数是否为素数。

样例输入：

5

样例输出：

yes

样例输入：

9

样例输出：

no

注意：是素数输出yes，不是素数输出no，其中yes和no均为小写。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      for(int i = 2; i * i <= n; i++) {
7          if(n % i == 0) {
8              cout << "no";
9              return 0;
10         }
11     }
```

```

12     cout << "yes";
13     return 0;
14 }

```

## ADV-92. 求最大公约数

编写一函数gcd，求两个正整数的最大公约数。

样例输入：

5 15

样例输出：

5

样例输入：

7 2

样例输出：

1

```

1  #include <iostream>
2  using namespace std;
3  int gcd(int a, int b) {
4      return b == 0 ? a : gcd(b, a%b);
5  }
6  int main() {
7      int a, b;
8      cin >> a >> b;
9      cout << gcd(a, b);
10     return 0;
11 }

```

## ADV-94. 复数归一化

编写函数Normalize，将复数归一化，即若复数为 $a+bi$ ，归一化结果为 $a/\sqrt{a^2+b^2} + i*b/\sqrt{a^2+b^2}$ 。使用结构体指针类型作为函数参数可能是必要的。其中实部和虚部由键盘输入，输出为归一化结果，如果归一化结果的实部或虚部为小数的要求保留一位小数。

样例输入：

3 4

样例输出：

0.6+0.8i

样例输入：

2 5

样例输出：

0.4+0.9i

分析：没什么好分析的耶。。

```
1  #include <iostream>
2  #include <cmath>
3  #include <iomanip>
4  using namespace std;
5  int main() {
6      double a, b;
7      double p, q;
8      cin >> a >> b;
9      p = a / sqrt(a * a + b * b);
10     q = b / sqrt(a * a + b * b);
11     cout << setprecision(1) << p;
12     if(q > 0) {
13         cout << "+";
14         cout << setprecision(1) << q;
15         cout.unsetf(ios_base::floatfield);
16         cout << "i";
17     } else if(q < 0) {
18         cout << setprecision(1) << q;
19         cout.unsetf(ios_base::floatfield);
20         cout << "i";
21     }
22     return 0;
23 }
```

## ADV-95. 字符串比较

独立实现标准字符串库的strcmp函数，即字符串比较函数，从键盘输入两个字符串，按字典序比较大小，前者大于后者输出1，前者小于后者输出-1，两者相等输出0。

样例输入：

apple one

样例输出：

-1

样例输入：

hello he

样例输出：

1

样例输入：

hello hello

样例输出：

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      string a, b;
5      cin >> a >> b;
6      int lena = a.length();
7      int lenb = b.length();
8      int len = lena > lenb ? lena : lenb;
9      for(int i = 0; i < len; i++) {
10         if(a[i] > b[i]) {
11             cout << "1";
12             return 0;
13         }
14         if(a[i] < b[i]) {
15             cout << "-1";
16             return 0;
17         }
18     }
19     if(lena == lenb) {
20         cout << "0";
21     } else if(lena > lenb) {
22         cout << "1";
23     } else {
24         cout << "-1";
25     }
26     return 0;
27 }

```

## ADV-96. 复数求和

从键盘读入n个复数（实部和虚部都为整数）用链表存储，遍历链表求出n个复数的和并输出。

样例输入：

3

3 4

5 2

1 3

样例输出：

9+9i

样例输入：

7

1 2



3 4

2 5

1 8

6 4

7 9

3 7

样例输出:

23+39i

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a = 0, b = 0;
5      int n;
6      cin >> n;
7      for(int i = 0; i < n; i++) {
8          int ta, tb;
9          cin >> ta >> tb;
10         a += ta;
11         b += tb;
12     }
13     cout << a;
14     if(b >= 0) {
15         cout << "+" << b << "i";
16     } else if(b < 0) {
17         cout << b << "i";
18     }
19     return 0;
20 }
```

## ADV-97. 十进制数转八进制数

编写函数，其功能为把一个十进制数转换为其对应的八进制数。程序读入一个十进制数，调用该函数实现数制转换后，输出对应的八进制数。

样例输入

9274

样例输出

22072

样例输入

18

样例输出

22

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      if (n == 0)
7          return 0;
8      int b[30];
9      int i = 0;
10     while (n != 0) {
11         b[i++] = n % 8;
12         n = n / 8;
13     }
14     for (int j = i - 1; j >= 0; j--) {
15         cout << b[j];
16     }
17     return 0;
18 }
```

## ADV-98. 约数个数

输入一个正整数N

样例输入

12

样例输出

6

样例说明

12的约数包括：1,2,3,4,6,12。共6个

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int N;
5      cin >> N;
6      int count = 0;
7      for (int i = 1; i <= N; i++) {
8          if (N % i == 0)
9              count++;
10     }
11     cout << count;
12 }
```

## ADV-99. 栅格打印问题

### 问题描述

编写一个程序，输入两个整数，作为栅格的高度和宽度，然后用“+”、“-”和“|”这三个字符来打印一个栅格。

输入格式：输入只有一行，包括两个整数，分别为栅格的高度和宽度。

输出格式：输出相应的栅格。

输入输出样例

### 样例输入

3 2

### 样例输出

+--++

| | |

+--++

| | |

+--++

| | |

+--++

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b;
5      cin >> a >> b;
6      if(a == 0 || b == 0)
7          return 0;
8      for(int i = 1; i <= a * 2 + 1; i++) {
9          if(i % 2 == 1) {
10             for(int j = 1; j <= b * 2 + 1; j++) {
11                 if(j % 2 == 1) {
12                     cout << "+";
13                 } else {
14                     cout << "-";
15                 }
16             }
17         } else {
18             for(int j = 1; j <= b * 2 + 1; j++) {
19                 if(j % 2 == 1) {
20                     cout << "|";
21                 } else {
22                     cout << " ";
```

```

23         }
24     }
25 }
26     cout << endl;
27 }
28     return 0;
29 }

```

## ADV-100. 第二大整数

### 问题描述

编写一个程序，读入一组整数（不超过20个），当用户输入0时，表示输入结束。

然后程序将从这组整数中，把第二大的那个整数找出来，并把它打印出来。

说明：（1）0表示输入结束，它本身并不计入这组整数中。（2）在这组整数中，既有正数，也可能有负数。（3）这组整数的个数不少于2个。

输入格式：输入只有一行，包括若干个整数，中间用空格隔开，最后一个整数为0。

输出格式：输出第二大的那个整数。

输入输出样例

### 样例输入

5 8 -12 7 0

### 样例输出

7

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  int cmp(int a, int b) {
6      return a > b;
7  }
8  int main() {
9      int a[20];
10     int count = 0;
11     for (int i = 0; i < 20; i++) {
12         cin >> a[i];
13         if (a[i] == 0) {
14             count = i;
15             break;
16         }
17     }
18     sort(a, a + count, cmp);
19     cout << a[1];

```

```
20     return 0;
21 }
```

## ADV-102. 单词个数统计

### 问题描述

编写一个程序，输入一个字符串（长度不超过80），然后统计出该字符串当中包含有多少个单词。例如：字符串“this is a book”当中包含有4个单词。

输入格式：输入一个字符串，由若干个单词组成，单词之间用一个空格隔开。

输入格式：

输出格式：输出一个整数，即单词的个数。

输出格式：

输入输出样例

用户输入数据样例：

this is a book

系统输出数据如下：

4

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string s;
5      int cnt = 0;
6      while(cin >> s) cnt++;
7      cout << cnt;
8      return 0;
9  }
```

## ADV-103. 逆序排列

### 问题描述

编写一个程序，读入一组整数（不超过20个），并把它们保存在一个整型数组中。

当用户输入0时，表示输入结束。然后程序将把这个数组中的值按逆序重新存放，并打印出来。

例如：假设用户输入了一组数据：7 19 -5 6 2 0，那么程序将会把前五个有效数据保存在一个数组中，即7 19 -5 6 2，然后把这个数组中的值按逆序重新存放，即变成了2 6 -5 19 7，然后把它们打印出来。

输入格式：输入只有一行，由若干个整数组成，中间用空格隔开，最末尾的整数为0。

输入格式：

输出格式：输出也只有一行，即逆序排列后的整数，中间用空格隔开，末尾没有空格。

输出格式：

输入输出样例

样例输入

7 19 -5 6 2 0

样例输出

2 6 -5 19 7

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a[20];
5      int count;
6      for (int i = 0; i < 20; i++) {
7          cin >> a[i];
8          if (a[i] == 0) {
9              count = i;
10             break;
11         }
12     }
13     for (int i = count - 1; i >= 0 ; i--)
14         cout << a[i] << " ";
15     return 0;
16 }
```

## ADV-104. 打水问题

问题描述

N个人要打水，有M个水龙头，第i个人打水所需时间为 $T_i$ ，请安排一个合理的方案使得所有人的等待时间之和尽量小。

输入格式

第一行两个正整数N M 接下来一行N个正整数 $T_i$ 。

$N, M \leq 1000$ ,  $T_i \leq 1000$

输出格式

最小的等待时间之和。（不需要输出具体的安排方案）

样例输入

7 3

3 6 1 4 2 5 7

样例输出

## 提示

一种最佳打水方案是，将N个人按照 $T_i$ 从小到大的顺序依次分配到M个龙头打水。

例如样例中， $T_i$ 从小到大排序为1, 2, 3, 4, 5, 6, 7，将他们依次分配到3个龙头，则去龙头一打水的为1, 4, 7；去龙头二打水的为2,5；去第三个龙头打水的为3,6。

第一个龙头打水的人总等待时间 =  $0 + 1 + (1 + 4) = 6$

第二个龙头打水的人总等待时间 =  $0 + 2 = 2$

第三个龙头打水的人总等待时间 =  $0 + 3 = 3$

所以总的等待时间 =  $6 + 2 + 3 = 11$

```

1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  using namespace std;
5  int main() {
6      int n, r;
7      scanf("%d %d", &n, &r);
8      int *a = new int[n];
9      for(int i = 0; i < n; i++) {
10         scanf("%d", &a[i]);
11     }
12     sort(a, a+n);
13     int ans = 0;
14     int cnt = n / r;
15     while(cnt-- > 0) {
16         for(int j = 0; j < cnt * r; j++)
17             ans += a[j];
18     }
19     for(int i = n/r*r; i < n; i++) {
20         for(int j = i % r; j < n/r*r; j += r)
21             ans += a[j];
22     }
23     cout << ans;
24     delete [] a;
25     return 0;
26 }
```

## ADV-105. 不同单词个数统计

### 问题描述

编写一个程序，输入一个句子，然后统计出这个句子当中不同的单词个数。例如：对于句子“one little two little three little boys”，总共有5个不同的单词：one, little, two, three, boys。

说明：（1）由于句子当中包含有空格，所以应该用gets函数来输入这个句子；（2）输入的句子当中只包含英文字符和空格，单词之间用一个空格隔开；（3）不用考虑单词的大小写，假设输入的都是小写字符；（4）句子长度不超过100个字符。

输入格式：输入只有一行，即一个英文句子。

输出格式：输出只有一行，是一个整数，表示句子中不同单词的个数。

输入输出样例

样例输入

one little two little three little boys

样例输出

5

```
1  #include <iostream>
2  #include <set>
3  using namespace std;
4  int main() {
5      string t;
6      set<string> s;
7      while(cin >> t) s.insert(t);
8      cout << s.size();
9      return 0;
10 }
```

## ADV-108. 分数统计

问题描述

2016.4.5已更新此题，此前的程序需要重新提交。

问题描述

给定一个百分制成绩T，将其划分为如下五个等级之一：

90~100为A，80~89为B，70~79为C，60~69为D，0~59为E

现在给定一个文件inp，文件中包含若干百分制成绩（成绩个数不超过100），请你统计五个等级段的人数，并找出人数最多的那个等级段，按照从大到小的顺序输出该段中所有人成绩（保证人数最多的等级只有一个）。要求输出到指定文件oup中。

输入格式

若干0~100的正整数，用空格隔开

输出格式

第一行为5个正整数，分别表示A,B,C,D,E五个等级段的人数

第二行一个正整数，表示人数最多的等级段中人数



接下来一行若干个用空格隔开的正整数，表示人数最多的那个等级中所有人的分数，按从大到小的顺序输出。

样例输入

100 80 85 77 55 61 82 90 71 60

样例输出

2 3 2 2 1

3

85 82 80

分析：它赖皮。。测试数据里面有分数的个数的值。。而测试样例里面没写。。。

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5  int cmp1(int a, int b) {return a > b; }
6  int main() {
7      vector<vector<int>> > v(5);
8      int s;
9      int n;
10     cin >> n;
11     for(int i = 0; i < n; i++) {
12         cin >> s;
13         if(s >= 90 && s <= 100) {
14             v[0].push_back(s);
15         } else if(s >= 80 && s <= 89) {
16             v[1].push_back(s);
17         } else if(s >= 70 && s <= 79) {
18             v[2].push_back(s);
19         } else if(s >= 60 && s <= 69) {
20             v[3].push_back(s);
21         } else {
22             v[4].push_back(s);
23         }
24     }
25     int maxn = 0, index = -1;
26     for(int i = 0; i < 5; i++) {
27         cout << v[i].size() << " ";
28         if(v[i].size() > maxn) {
29             maxn = v[i].size();
30             index = i;
31         }
32     }
33     cout << endl << maxn << endl;
34     sort(v[index].begin(), v[index].end(), cmp1);
35     for(int i = 0; i < v[index].size(); i++) {
```

```

36         cout << v[index][i] << " ";
37     }
38     return 0;
39 }

```

## ADV-109. 征税程序

### 问题描述

税务局希望你帮他们编写一个征税程序，该程序的功能是：首先输入某公司的年销售额sale和税率rate，然后程序将计算出相应的税额tax，并把它显示在屏幕上。计算公式是：

$\text{tax} = \text{sale} * \text{rate}$ 。

输入格式：输入只有一行，包括两个数据，即年销售额和税率。

输出格式：输出只有一行，包括一个实数，即相应的税额，保留到小数点后两位。

输入输出样例

样例输入

50000.5 0.1

样例输出

5000.50

```

1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main() {
5      double sale, rate;
6      cin >> sale >> rate;
7      double tax = sale * rate;
8      printf("%.2f", tax);
9      return 0;
10 }

```

## ADV-110. 温度转换

### 问题描述

编写一个程序，输入一个摄氏温度，输出相应的华氏温度。在输出时，保留小数点后面两位。

输入格式：输入只有一个整数，即摄氏温度。

输出格式：输出只有一实数，即相应的华氏温度。

输入输出样例

样例输入

35

样例输出

95.00

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main() {
5      int a;
6      cin >> a;
7      double b = a * 1.8 + 32;
8      printf("%.2f", b);
9      return 0;
10 }
```

## ADV-111. Quadratic Equation

问题描述

求解方程 $ax^2+bx+c=0$ 的根。要求 $a, b, c$ 由用户输入，并且可以为任意实数。

输入格式：输入只有一行，包括三个系数，之间用空格隔开。

输出格式：输出只有一行，包括两个根，大根在前，小根在后，无需考虑特殊情况，保留小数点后两位。

输入输出样例

样例输入

2.5 7.5 1.0

样例输出

-0.14 -2.86

分析：1.若  $x_1 < x_2$ ，且  $f(x_1) \cdot f(x_2) < 0$ ，则在  $(x_1, x_2)$  之间一定有一个根

2.找到一个区间内的根，二分逼近，找到误差很小的近似根，可认为是根

```
1  #include <cmath>
2  #include <iostream>
3  using namespace std;
4  double a, b, c;
5  double f(double x) {
6      return a * x * x + b * x + c;
7  }
8  double find(double x, double y) {
9      if (y - x < 0.000001) return x;
10     double mid = (x + y) / 2;
11     if (f(mid) == 0) return mid;
12     else if (f(mid) * f(x) > 0) return find(mid, y);
13     else return find(x, mid);
14 }
15 int main() {
```

```

16     cin >> a >> b >> c;
17     double mid = -1 * b / (2 * a);
18     double ans1 = find(-9999999, mid);
19     double ans2 = find(mid, 99999999);
20     if (abs(ans1 - 0) < 0.0001) ans1 = 0;
21     if (abs(ans2 - 0) < 0.0001) ans2 = 0;
22     printf("%.2f %.2f", ans2, ans1);
23     return 0;
24 }

```

## ADV-112. c++\_ch02\_01

编写一个程序，利用强制类型转换打印元音字母大小写10种形式的ASCII码。

输出的顺序为：大写的字母A, E, I, O, U的ASCII码，小写的字母a, e, i, o, u的ASCII码

所有的ASCII码都用十进制表示.输出10行,每行一个ASCII码，最后输出一个空行。

分析：static\_cast<new type> (expression) 函数能够将括号中的表达式转换成new\_type类型的数值，比如static\_cast<int> (c[i])能将c[i]以int类型返回～

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      char c[10] = {'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'};
5      for (int i = 0; i < 10; i++)
6          cout << static_cast<int> (c[i]) << endl;
7      return 0;
8  }

```

## ADV-113. c++\_ch02\_02

使用Switch语句编写一个模拟简单计算器的程序。依次输入两个整数和一个字符，并用空格隔开。如果该字符是一个“+”，则打印和；如果该字符是一个“-”，则打印差；如果该字符是一个“\*”，则打印积；如果该字符是“/”，则打印商；如果该字符是一个“%”，则打印余数。打印结果后输出一个空行。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b, ans;
5      char c;
6      cin >> a >> b >> c;
7      if (c == '+') ans = a + b;
8      else if (c == '-') ans = a - b;
9      else if (c == '*') ans = a * b;
10     else if (c == '%') ans = a % b;
11     else if (c == '/') ans = a / b;
12     cout << ans;
13     return 0;
14 }

```

## ADV-117. 进制转换

### 问题描述

程序提示用户输入三个字符，每个字符取值范围是0-9，A-F。然后程序会把这三个字符转化为相应的十六进制整数，并分别以十六进制，十进制，八进制输出。

输入格式：输入只有一行，即三个字符。

输出格式：输出只有一行，包括三个整数，中间用空格隔开。

输入输出样例

样例输入

FFF

样例输出

FFF 4095 7777

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n = 0;
5      char c1, c2, c3;
6      string s(3, '0'), ans8;
7      scanf("%c %c %c", &c1, &c2, &c3);
8      s[0] = c1;
9      s[1] = c2;
10     s[2] = c3;
11     for (int i = 0; i < s.length(); i++) {
12         if (s[i] >= '0' && s[i] <= '9') n = n * 16 + s[i] - '0';
13         else n = n * 16 + s[i] - 'A' + 10;
14     }
15     if (s[0] == '0') s = s.substr(1, s.length() - 1);
16     if (s[0] == '0') s = s.substr(1, s.length() - 1);
17     cout << s << " " << n << " ";
18     while (n) {
19         ans8 = char(n % 8 + '0') + ans8;
20         n /= 8;
21     }
22     if (ans8 == "") ans8 = "0";
23     cout << ans8;
24     return 0;
25 }
```

## ADV-118. 3-2字符串输入输出函数

### 描述

编写函数GetReal和GetString，在main函数中分别调用这两个函数。在读入一个实数和一个字符串后，将读入的结果依次用printf输出。

两次输入前要输出的提示信息分别是“please input a number:\n”和“please input a string:\n”

样例输入

9.56

hello

样例输出

please input a number:

please input a string:

9.56

hello

```
1  #include <iostream>
2  using namespace std;
3  void GetReal() {
4      cout << "please input a number:\n";
5  }
6
7  void GetString() {
8      cout << "please input a string:\n";
9  }
10
11 int main() {
12     GetReal();
13     GetString();
14     string n, s;
15     cin >> n >> s;
16     cout << n << endl << s;
17     return 0;
18 }
```

## ADV-119. 6-9删除数组中的0元素

编写函数CompactIntegers，删除数组中所有值为0的元素，其后元素向数组首端移动。注意，CompactIntegers函数需要接收数组及其元素个数作为参数，函数返回值应为删除操作执行后数组的新元素个数。

输入时首先读入数组长度，再依次读入每个元素。

将调用此函数后得到的数组和函数返回值输出。

样例输入

7

2 0 4 3 0 0 5

样例输出

2 4 3 5

4

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int cnt = 0;
7      for(int i = 0; i < n; i++) {
8          int t;
9          cin >> t;
10         if(t != 0) {
11             cout << t << " ";
12             cnt++;
13         }
14     }
15     cout << endl << cnt;
16     return 0;
17 }
```

## ADV-120. 6-17复数四则运算

设计复数库，实现基本的复数加减乘除运算。

输入时只需分别键入实部和虚部，以空格分割，两个复数之间用运算符分隔；输出时按a+bi的格式在屏幕上打印结果。参加样例输入和样例输出。

注意考虑特殊情况，无法计算时输出字符串“error”。

样例输入

2 4 \* -3 2

样例输出

-14-8i

样例输入

3 -2 + -1 3

样例输出

2+1i

分析：除法的时候有点麻烦，结果要变成浮点型，因为测试用例的答案是统一输出一位有效数字的。。所以0.123输出0.1 1.000输出1。。。我用了cout的setprecision控制输出的有效数字个数~

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
```

```

4
5 void add(int a, int b, int m, int n);
6 void min(int a, int b, int m, int n);
7 void multi(int a, int b, int m, int n);
8 void divl(int a, int b, int m, int n);
9
10 int main() {
11     int a,b,m,n;
12     char c;
13     cin >> a >> b;
14     cin >> c;
15     cin >> m >> n;
16     switch(c) {
17         case '+': add(a, b, m, n); break;
18         case '-': min(a, b, m, n); break;
19         case '*': multi(a, b, m, n); break;
20         case '/': divl(a, b, m, n); break;
21     }
22     return 0;
23 }
24
25 void add(int a, int b, int m, int n) {
26     int l, r;
27     l = a + m;
28     r = b + n;
29     if (l != 0){
30         if (r != 0) {
31             if (r > 0)
32                 cout << l << "+" << r << "i";
33             else
34                 cout << l << r << "i";
35         }
36         else {
37             cout << l;
38         }
39     }
40     else {
41         if (r != 0) {
42             cout << r << "i";
43         }
44         else {
45             cout << 0;
46         }
47     }
48 }
49
50 void min(int a, int b, int m, int n) {
51     int l, r;
52     l = a - m;

```



```

53     r = b - n;
54     if (l != 0) {
55         if (r != 0) {
56             if (r > 0)
57                 cout << l << "+" << r << "i";
58             else
59                 cout << l << r << "i";
60         }
61         else {
62             cout << l;
63         }
64     }
65     else {
66         if (r != 0) {
67             cout << r << "i";
68         }
69         else {
70             cout << 0;
71         }
72     }
73 }
74
75 void multi(int a, int b, int m, int n) {
76     int l, r;
77     l = a * m - b * n;
78     r = a * n + b * m;
79     if (l != 0) {
80         if (r != 0) {
81             if (r > 0)
82                 cout << l << "+" << r << "i";
83             else
84                 cout << l << r << "i";
85         }
86         else {
87             cout << l;
88         }
89     }
90     else {
91         if (r != 0) {
92             cout << r << "i";
93         }
94         else {
95             cout << 0;
96         }
97     }
98 }
99
100 void div1(int a, int b, int m, int n) {
101     if (m == 0 && n == 0) {

```

```

102         cout << "error";
103     } else {
104         double l, r;
105         l = (double)(a * m + b * n) / (m * m + n * n);
106         r = (double)(b * m - a * n) / (m * m + n * n);
107
108         if (l != 0) {
109             if (r != 0) {
110                 if (r > 0) {
111                     cout << setprecision(1) << l << "+" << r << endl;
112                 }
113                 else {
114                     cout << setprecision(1) << l << r << "i" << endl;
115                 }
116             }
117             else {
118                 cout << setprecision(1) << l << endl;
119             }
120         } else {
121             if (r != 0) {
122                 cout << setprecision(1) << r << "i" << endl;
123             }
124             else {
125                 cout << 0;
126             }
127         }
128     }
129 }

```

## ADV-121. 高精度加法

### 问题描述

在C/C++语言中，整型所能表示的范围一般为-231到231（大约21亿），即使long long型，一般也只能表示到-263到263。要想计算更加规模的数，就要用软件来扩展了，比如用数组或字符串来模拟更多规模的数及共运算。

现在输入两个整数，请输出它们的和。

### 输入格式

两行，每行一个整数，每个整数不超过1000位

### 输出格式

一行，两个整数的和。

### 样例输入

```

15464315464465465
482321654151

```

### 样例输出

15464797786119616

数据规模和约定

每个整数不超过1000位

分析：1.模拟竖式加法，依次从往左加

2.如果刚开始两位数字位数不一样，短的用0补上，最后一次加法，如果有进位也要加上～

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  string add(string s1, string s2) {
5      int len1 = s1.length(), len2 = s2.length();
6      if (len1 < len2) {
7          string t(len2 - len1, '0');
8          s1 = t + s1;
9      } else if (len2 < len1) {
10         string t(len1 - len2, '0');
11         s2 = t + s2;
12     }
13     string ans = s1;
14     int car = 0;
15     for (int i = s1.length() - 1; i >= 0; i--) {
16         ans[i] = (s1[i] - '0' + s2[i] - '0' + car) % 10 + '0';
17         car = (s1[i] - '0' + s2[i] - '0' + car) / 10;
18     }
19     if (car) ans = (char) (car + '0') + ans;
20     return ans;
21 }
22 int main() {
23     string s1, s2;
24     cin >> s1 >> s2;
25     cout << add(s1, s2);
26     return 0;
27 }
```

## ADV-126. 扫雷

问题描述

扫雷游戏你一定玩过吧！现在给你若干个 $n \times m$ 的地雷阵，请你计算出每个矩阵中每个单元格相邻单元格内地雷的个数，每个单元格最多有8个相邻的单元格。 $0 < n, m \leq 100$

输入格式

输入包含若干个矩阵，对于每个矩阵，第一行包含两个整数 $n$ 和 $m$ ，分别表示这个矩阵的行数和列数。接下来 $n$ 行每行包含 $m$ 个字符。安全区域用'.'表示，有地雷区域用'\*'表示。当 $n=m=0$ 时输入结束。

输出格式

对于第*i*个矩阵，首先在单独的一行里打印序号：“Field #i:”，接下来的*n*行中，读入的‘.’应被该位置周围的地雷数所代替。输出的每两个矩阵必须用一个空行隔开。

样例输入

```
4 4
*...
....
.*..
....
3 5
**...
.....
.*...
0 0
```

样例输出

```
Field #1:
*100
2210
1*10
1110
```

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, m, ans[150][150], cnt = 0;
5      char a[150][150];
6      while (cin >> n >> m) {
7          fill(a[0], a[0] + 150 * 150, '.');
8          fill(ans[0], ans[0] + 150 * 150, -1);
9          if (n == 0 && m == 0) break;
10         printf("Field #%d:\n", ++cnt);
11         scanf("%d", n);
12         for (int i = 1; i <= n; i++)
13             for (int j = 1; j <= m; j++)
14                 scanf("%c ", &a[i][j]);
15         for (int i = 1; i <= n; i++) {
16             for (int j = 1; j <= m; j++) {
17                 if (a[i][j] == '.') {
18                     int sum = 0;
19                     if (a[i][j + 1] == '*') sum++;
20                     if (a[i][j - 1] == '*') sum++;
21                     if (a[i + 1][j] == '*') sum++;
22                     if (a[i - 1][j] == '*') sum++;
23                     if (a[i + 1][j + 1] == '*') sum++;
24                     if (a[i + 1][j - 1] == '*') sum++;
25                     if (a[i - 1][j + 1] == '*') sum++;
26                     if (a[i - 1][j - 1] == '*') sum++;
```

```

27         a[i][j] = sum + '0';
28     }
29     cout << a[i][j];
30 }
31 cout << endl;
32 }
33 cout << endl;
34 }
35 return 0;
36 }

```

## ADV-127. 日期计算

### 问题描述

已知2011年11月11日是星期五，问YYYY年MM月DD日是星期几？注意考虑闰年的情况。尤其是逢百年不闰，逢400年闰的情况。

### 输入格式

输入只有一行

YYYY MM DD

### 输出格式

输出只有一行

W

### 数据规模和约定

1599 <= YYYY <= 2999

1 <= MM <= 12

1 <= DD <= 31，且确保测试样例中YYYY年MM月DD日是一个合理日期

1 <= W <= 7，分别代表周一到周日

### 样例输入

2011 11 11

### 样例输出

5

```

1  #include <iostream>
2  using namespace std;
3
4  int isleapyear(int year);
5  int days(int year, int month, int day);
6  int distance(int year, int month, int day);
7

```

```

8  int main() {
9      int year, month, day;
10     cin >> year >> month >> day;
11     int distance1 = distance(year, month, day);
12     if (year >= 2012) {
13         switch(distance1) {
14             case 0: cout << 6; break;
15             case 1: cout << 7; break;
16             case 2: cout << 1; break;
17             case 3: cout << 2; break;
18             case 4: cout << 3; break;
19             case 5: cout << 4; break;
20             case 6: cout << 5; break;
21         }
22     } else {
23         switch(distance1) {
24             case 0: cout << 6; break;
25             case 1: cout << 5; break;
26             case 2: cout << 4; break;
27             case 3: cout << 3; break;
28             case 4: cout << 2; break;
29             case 5: cout << 1; break;
30             case 6: cout << 7; break;
31         }
32     }
33     return 0;
34 }
35
36 int isleapyear(int year) {
37     if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)
38         return 1;
39     else
40         return 0;
41 }
42
43 int days(int year, int month, int day) {
44     int sum = 0;
45     if (month <= 2) {
46         if (month == 1)
47             sum = day;
48         else
49             sum = 31 + day;
50     } else {
51         switch(month) {
52             case 3: sum = 59 + day; break;
53             case 4: sum = 90 + day; break;
54             case 5: sum = 120 + day; break;
55             case 6: sum = 151 + day; break;
56             case 7: sum = 181 + day; break;

```

```

57         case 8: sum = 212 + day; break;
58         case 9: sum = 243 + day; break;
59         case 10: sum = 273 + day; break;
60         case 11: sum = 304 + day; break;
61         case 12: sum = 334 + day; break;
62     }
63     if(isleapyear(year) == 1) {
64         sum = sum + 1;
65     }
66 }
67 return sum;
68 }
69
70 int distance(int year, int month, int day) {
71     int count = 0;
72     int distance1 = 0;
73     if(year >= 2012) {
74         distance1 = 365 * (year - 2012) + days(year, month, day);
75         for (int i = 2012; i < year; i++) {
76             if(isleapyear(i) == 1)
77                 count++;
78         }
79         distance1 = distance1 + count;
80     } else {
81         distance1 = 365 * (2011 - year) + 365 - days(year, month, day);
82         if (isleapyear(year) == 1)
83             distance1 = distance1 + 1;
84         for (int i = year + 1; i <= 2011; i++) {
85             if(isleapyear(i) == 1)
86                 count++;
87         }
88         distance1 = distance1 + count;
89     }
90     distance1 = distance1 % 7;
91     return distance1;
92 }

```

## ADV-130. 色盲的民主

### 色盲的民主

#### 问题描述

n个色盲聚在一起，讨论一块布的颜色。尽管都是色盲，却盲得各不相同。每个人都有自己的主张，争论不休。最终，他们决定采取民主投票的方式决定布的颜色，不管布同不同意。某种颜色用字符串表示(字符串为颜色单词或词组，也就是可能有被空格隔开的两个单词组成的颜色词组)，只要字符串不同，程序即判断颜色不同。现在给出这n个人所选择的颜色，输出最有可能的颜色（也就是获得投票最多的颜色），如果有多个颜色获得了最多的投票，则将它们按字典序分行全部输出。

#### 输入格式

第一行一个正整数n，表示色盲的人数

接下来n行，每行一句话

输出格式

若干行，获得投票最多的颜色，按字典序输出

样例输入

```
5
red
blue
black
black
blue
```

样例输出

```
black
blue
```

数据规模和约定

$n \leq 1000$

颜色单词最多20个字符，只包含小写字母或者空格

注

对于char s[20]，由于cin >> s是读取到空格处便会结束，也就是对于light red，用cin只能输入light。如果要整个输入一行，则使用cin.getline(s, 20)，其中20为这一行的最大长度，也就是你的s的容量，如果容量为30，则cin.getline(s, 30)。

另外，你在cin>>n以后cin.getline(s,30)应该会得到一个空字符串，这是因为整数n后面的换行符还未被输入。

```
1  #include <iostream>
2  #include <map>
3  #include <cmath>
4  using namespace std;
5  int main() {
6      int n, maxn = 0;
7      string s;
8      map<string, int> m;
9      cin >> n;
10     for (int i = 0; i < n; i++) {
11         getline(cin, s);
12         m[s]++;
13         maxn = max(maxn, m[s]);
14     }
15     for (map<string, int>::iterator i = m.begin(); i != m.end(); i++) {
16         if (i->second == maxn)
17             cout << i->first << endl;
18     }
```



```
19     return 0;
20 }
```

## ADV-131. 选择排序

### 问题描述

排序，顾名思义，是将若干个元素按其大小关系排出一个顺序。形式化描述如下：有 $n$ 个元素 $a[1], a[2], \dots, a[n]$ ，从小到大排序就是将它们排成一个新顺序 $a[i[1]] < a[i[2]] < \dots < a[i[n]]$

$i[k]$ 为这个新顺序。

选择排序的思想极其简单，每一步都把一个最小元素放到前面，如果有多个相等的最小元素，选择排位较靠前的放到当前头部。还是那个例子：{3 1 5 4 2}：

第一步将1放到开头（第一个位置），也就是交换3和1，即 $\text{swap}(a[0], a[1])$ 得到{1 3 5 4 2}

第二步将2放到第二个位置，也就是交换3和2，即 $\text{swap}(a[1], a[4])$ 得到{1 2 5 4 3}

第三步将3放到第三个位置，也就是交换5和3，即 $\text{swap}(a[2], a[4])$ 得到{1 2 3 4 5}

第四步将4放到第四个位置，也就是交换4和4，即 $\text{swap}(a[3], a[3])$ 得到{1 2 3 4 5}

第五步将5放到第五个位置，也就是交换5和5，即 $\text{swap}(a[4], a[4])$ 得到{1 2 3 4 5}

输入 $n$ 个整数，输出选择排序的全过程。

要求使用递归实现。

第一行一个正整数 $n$ ，表示元素个数

第二行为 $n$ 个整数，以空格隔开

### 输出格式

共 $n$ 行，每行输出第 $n$ 步选择时交换哪两个位置的下标，以及交换得到的序列，格式：

$\text{swap}(a[i], a[j]): a[0] \dots a[n-1]$

$i$ 和 $j$ 为所交换元素的下标，下标从0开始，最初元素顺序按输入顺序。另外请保证 $i \leq j$

$a[0] \dots a[n-1]$ 为交换后的序列，元素间以一个空格隔开

### 样例输入

5

4 3 1 1 2

### 样例输出

$\text{swap}(a[0], a[2]): 1\ 3\ 4\ 1\ 2$

$\text{swap}(a[1], a[3]): 1\ 1\ 4\ 3\ 2$

$\text{swap}(a[2], a[4]): 1\ 1\ 2\ 3\ 4$

$\text{swap}(a[3], a[3]): 1\ 1\ 2\ 3\ 4$

swap(a[4], a[4]):1 1 2 3 4

数据规模和约定

$n \leq 100$

整数元素在int范围内

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int *a = new int[n];
7      for(int i = 0; i < n; i++) {
8          cin >> a[i];
9      }
10     for(int i = 0; i < n; i++) {
11         int k = i;
12         for(int j = i + 1; j < n; j++) {
13             if(a[j] < a[k])
14                 k = j;
15         }
16         swap(a[i], a[k]);
17         printf("swap(a[%d], a[%d]):", i, k);
18         for(int m = 0; m < n; m++) {
19             printf("%d ", a[m]);
20         }
21         cout << endl;
22     }
23     delete [] a;
24     return 0;
25 }
```

## ADV-132. 笨小猴

问题描述

笨小猴的词汇量很小，所以每次做英语选择题的时候都很头疼。但是他找到了一种方法，经试验证明，用这种方法去选择选项的时候选对的几率非常大！

这种方法的具体描述如下：假设maxn是单词中出现次数最多的字母的出现次数，minn是单词中出现次数最少的字母的出现次数，如果maxn-minn是一个质数，那么笨小猴就认为这是个Lucky Word，这样的单词很可能就是正确的答案。

输入格式

输入文件只有一行，是一个单词，其中只可能出现小写字母，并且长度小于100。

输出格式

输出文件共两行，第一行是一个字符串，假设输入的单词是Lucky Word，那么输出“Lucky Word”，否则输出“No Answer”；第二行是一个整数，如果输入单词是Lucky Word，输出maxn-minn的值，否则输出0。

样例输入

error

样例输出

Lucky Word

2

样例说明

单词error中出现最多的字母r出现了3次，出现次数最少的字母出现了1次， $3-1=2$ ，2是质数。

样例输入

olympic

样例输出

No Answer

0

样例说明

单词olympic中所有字母都只出现了1次， $1-1=0$ ，0不是质数。

```
1  #include <iostream>
2  #include <cctype>
3  using namespace std;
4  bool isprime(int n) {
5      if(n <= 1)
6          return false;
7      for(int i = 2; i * i <= n; i++) {
8          if(n % i == 0)
9              return false;
10     }
11     return true;
12 }
13
14 int main() {
15     int maxn = -1, minn = 99999999;
16     int a[26] = {0};
17     string s;
18     cin >> s;
19     for(int i = 0; i < s.length(); i++) {
20         char t = toupper(s[i]);
21         a[t-'A']++;
22     }
```

```

23     for(int i = 0; i < 26; i++) {
24         maxn = maxn > a[i] ? maxn : a[i];
25         if(a[i] != 0)
26             minn = minn < a[i] ? minn : a[i];
27     }
28     if(isprime(maxn-minn)) {
29         cout << "Lucky Word" << endl << maxn - minn;
30     } else {
31         cout << "No Answer" << endl << 0;
32     }
33     return 0;
34 }

```

## ADV-133. 彩票

### 问题描述

为丰富男生节活动，贵系女生设置彩票抽奖环节，规则如下：

- 1、每张彩票上印有7个各不相同的号码，且这些号码的取值范围为[1, 33]；
- 2、每次在兑奖前都会公布一个由七个互不相同的号码构成的中奖号码；
- 3、共设置7个奖项，特等奖和一等奖至六等奖。兑奖规则如下：

特等奖：要求彩票上的7个号码都出现在中奖号码中；

一等奖：要求彩票上的6个号码出现在中奖号码中；

二等奖：要求彩票上的5个号码出现在中奖号码中；

.....

六等奖：要求彩票上的1个号码出现在中奖号码中；

注：不考虑号码出现的顺序，例如若中奖号码为23 31 1 14 19 17 18，则彩票12 8 9 23 1 16 7由于其中有两个号码（23和1）出现在中奖号码中，所以该彩票中了五等奖。

现已知中奖号码和李华买的若干彩票的号码，请你写一个程序判断他的彩票中奖情况。

### 输入格式

第一行一个正整数n，表示彩票数量，第二行7个整数，表示中奖号码，下面n行每行7个整数，描述n张彩票。

### 输出格式

7个空格隔开的数字，第1个数字表示特等奖的中奖张数，第2个数字表示一等奖的中奖张数，第3个数字表示二等奖的中奖张数.....第7个数字表示六等奖的中奖张数。

### 样例输入

```

3
1 2 3 4 5 6 7
11 12 13 14 15 16 17
12 13 14 15 16 17 18
8 7 10 9 31 30 29

```

### 样例输出

```

0 0 0 0 0 0 1

```

## 数据规模和约定

30%的数据 $n \leq 100$ ;

70%的数据 $n \leq 1000$ ;

100%的数据 $n \leq 100000$ 。

\*\*\*\*\*提示：数组定义为全局变量，可以分配更多内存。\*\*\*\*\*

分析：1.m数组标记中奖号码，mt数组标记彩票号码

2.如果一个号码在中奖号码中出现多次，且彩票中也出现多次，要取他们的最小值，不能当成只中了一个号码哦~

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, m[34] = {0}, a[10] = {0};
5      cin >> n;
6      for (int i = 0; i < 7; i++) {
7          int t;
8          cin >> t;
9          m[t]++;
10     }
11     while (n--) {
12         int mt[34] = {0}, sum = 0;
13         for (int i = 0; i < 7; i++) {
14             int t;
15             cin >> t;
16             mt[t]++;
17         }
18         for (int i = 1; i <= 33; i++)
19             sum += min(mt[i], m[i]);
20         a[7 - sum]++;
21     }
22     for (int i = 0; i < 7; i++) {
23         cout << a[i] << " ";
24     }
25     return 0;
26 }
```

## ADV-134. 校门外的树

### 问题描述

某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数轴上的每个整数点，即0，1，2，……，L，都种有一棵树。

由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树都移走后，马路上还有多少棵树。

### 输入格式

输入的第一行有两个整数L ( $1 \leq L \leq 10000$ ) 和 M ( $1 \leq M \leq 100$ ) , L代表马路的长度, M代表区域的数目, L和M之间用一个空格隔开。接下来的M行每行包含两个不同的整数, 用一个空格隔开, 表示一个区域的起始点和终止点的坐标。

### 输出格式

输出包括一行, 这一行只包含一个整数, 表示马路上剩余的树的数目。

### 样例输入

500 3

150 300

100 200

470 471

### 样例输出

298

### 数据规模和约定

对于20%的数据, 区域之间没有重合的部分;

对于其它的数据, 区域之间有重合的情况。

分析: 用与l+1等长的数组标记该区域是否有树

```
1  #include <iostream>
2  #include <cstdio>
3  int book[10001];
4  using namespace std;
5  int main() {
6      int l, m;
7      scanf("%d %d", &l, &m);
8      for(int i = 0; i < m; i++) {
9          int a, b;
10         scanf("%d %d", &a, &b);
11         for(int j = a; j <= b; j++)
12             book[j] = 1;
13     }
14     int cnt = 0;
15     for(int i = 0; i <= l; i++) {
16         if(book[i] == 0)
17             cnt++;
18     }
19     printf("%d", cnt);
20     return 0;
21 }
```

## ADV-135. 三角形面积

### 问题描述

由三角形的三边长，求其面积。

提示：由三角形的三边a,b,c求面积可以用如下的公式：

$s = (a+b+c) / 2$

面积=

$$\sqrt{s(s-a)(s-b)(s-c)}$$

### 输入格式

由空格分开的三个整数。

### 输出格式

一个实数，保留两位小数。

### 样例输入

3 4 5

### 样例输出

6.00

### 数据规模和约定

输入的三条边一定能构成三角形，不用进行判定。a,b,c小于1000

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      int a, b, c;
6      cin >> a >> b >> c;
7      double s = (a + b + c) / 2.0;
8      printf("%.2f", sqrt(s * (s - a) * (s - b) * (s - c)));
9      return 0;
10 }
```

## ADV-136. 大数加法

### 问题描述

输入两个正整数a,b，输出a+b的值。

### 输入格式

两行，第一行a，第二行b。a和b的长度均小于1000位。

输出格式

一行，a+b的值。

样例输入

4

2

样例输出

6

分析：用两个指针一起从后往前加两个字符串，用carry变量表示进位，如果最后还有进位就还要再加1。

```
1  include <iostream>
2  using namespace std;
3  int main() {
4      string a, b;
5      cin >> a >> b;
6      char ans[1001];
7      int p = a.length() - 1;
8      int q = b.length() - 1;
9      int carry = 0;
10     int i = 0;
11     while(p != -1 && q != -1) {
12         int ta = a[p] - '0';
13         int tb = b[q] - '0';
14         if(ta + tb + carry >= 10) {
15             ans[i++] = (char)(ta + tb + carry - 10 + '0');
16             carry = 1;
17         } else {
18             ans[i++] = (char)(ta + tb + carry + '0');
19             carry = 0;
20         }
21         p--;
22         q--;
23     }
24     while(p != -1) {
25         int ta = a[p] - '0';
26         if(ta + carry >= 10) {
27             ans[i++] = (char)(ta + carry - 10 + '0');
28             carry = 1;
29         } else {
30             ans[i++] = (char)(ta + carry + '0');
31             carry = 0;
32         }
33         p--;
```



```

34     }
35     while(q != -1) {
36         int tb = b[q] - '0';
37         if(tb + carry >= 10) {
38             ans[i++] = (char)(tb + carry - 10 + '0');
39             carry = 1;
40         } else {
41             ans[i++] = (char)(tb + carry + '0');
42             carry = 0;
43         }
44         q--;
45     }
46     if(carry == 1) {
47         ans[i] = '1';
48     }
49     for(; i >= 0; i--) {
50         cout << ans[i];
51     }
52     return 0;
53 }

```

## ADV-140. 开灯游戏

### 问题描述

有9盏灯与9个开关，编号都是1~9。

每个开关能控制若干盏灯，按下一次会改变其控制的灯的状态(亮的变成不亮，不亮变成亮的)。

具体如下：

第一个开关控制第二，第四盏灯；

第二个开关控制第一，第三，第五盏灯；

第三个开关控制第二，第六盏灯；

第四个开关控制第一，第五，第七盏灯；

第五个开关控制第二，第四，第六，第八盏灯；

第六个开关控制第三，第五，第九盏灯；

第七个开关控制第四，第八盏灯；

第八个开关控制第五，第七，第九盏灯；

第九个开关控制第六，第八盏灯。

开始时所有灯都是熄灭的，开关是关闭着的。要求按下若干开关后，使得只有4盏灯亮着。

### 输出格式

输出所有可能的方案，每行一个方案，每一行有9个字符，从左往右第i个字符表示第i个开关的状态("0"表示关闭，"1"表示打开)，按字典序输出。下面的样例输出只是部分方案。

### 样例输出

```

000001011
000001110
000001111

```

分析：0~2<sup>9</sup>的数字换算成2进制，得到开关的每一种状态，每种判断一次即可～

```

1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4  void f(string s) {
5      int a[11], sum = 0;
6      memset(a, -1, sizeof(a));
7      if (s[0] == '1') a[2] *= -1, a[4] *= -1;
8      if (s[1] == '1') a[1] *= -1, a[3] *= -1, a[5] *= -1;
9      if (s[2] == '1') a[2] *= -1, a[6] *= -1;
10     if (s[3] == '1') a[1] *= -1, a[5] *= -1, a[7] *= -1;
11     if (s[4] == '1') a[2] *= -1, a[4] *= -1, a[6] *= -1, a[8] *= -1;
12     if (s[5] == '1') a[3] *= -1, a[5] *= -1, a[9] *= -1;
13     if (s[6] == '1') a[4] *= -1, a[8] *= -1;
14     if (s[7] == '1') a[5] *= -1, a[7] *= -1, a[9] *= -1;
15     if (s[8] == '1') a[6] *= -1, a[8] *= -1;
16     for (int i = 1; i <= 9; i++)
17         if (a[i] == 1) sum++;
18     if (sum == 4) cout << s << endl;
19 }
20 int main() {
21     for (int i = 0; i < 512; i++) {
22         int n = i;
23         string s;
24         while (n) {
25             s = (char) (n % 2 + '0') + s;
26             n /= 2;
27         }
28         string t(9 - s.length(), '0');
29         s = t + s;
30         f(s);
31     }
32     return 0;
33 }

```

## ADV-141. 判断名次

### 问题描述

某场比赛过后，你想要知道A~E五个人的排名是什么，于是要求他们每个人说了一句话。（经典的开头.....-\_-!）得了第1名的人23，说了假话；得了第5名的人不好意思，也说了假话；为了使求解问题简单，第3名同样说了假话。（奇数名次说假话）

### 输入格式

共5行，各行依次表示A~E说的话。

每行包含一个形如“A>=3”的名次判断，即一个大写字母+关系运算符+一个数字，不包含空格。

大写字母A~E，关系运算<、<=、=、>=、>、!=，数字1~5。注意：等于是“=”不是“==”！

### 输出格式

可能有多解，请按照字典序输出排名序列，每个解一行  
最后一行输出解的数量

样例输入

A=2  
D=5  
E>3  
A>2  
B!=1

样例输出

ACDEB  
AECBD  
BADCE  
BCADE  
BDACE  
CEADB  
CEBDA  
7

分析：1.用全排列公式，枚举除所有可能的排序，检查每个排序是否符合要求  
2.检查每种名次，如果出现了奇数名次的人说了真话，该序列不符合，如果偶数名次的人说了假话，也不符合～

```
1  #include <iostream>
2  #include <algorithm>
3  #include <vector>
4  #include <queue>
5  #include <cmath>
6  using namespace std;
7  string ranks = "ABCDE";
8  vector<string> say(5);
9  int cnt;
10 bool check(string s) {
11     int hash[150] = {0};
12     for (int i = 0; i < 5; i++)
13         hash[s[i]] = i + 1;
14     for (int i = 0; i < 5; i++) {
15         string sayt = say[i];
16         int len = sayt.length();
17         if (hash['A' + i] % 2 == 1) {
18             if (sayt[1] == '=' && hash[sayt[0]] == sayt[len - 1] - '0')
19                 return false;
20             if (sayt[1] == '!' && hash[sayt[0]] != sayt[len - 1] - '0')
21                 return false;
22             if (len == 3 && sayt[1] == '<' && hash[sayt[0]] < sayt[len - 1] - '0') return false;
23         }
24     }
25     return true;
26 }
```

```

21         if (len == 3 && sayt[1] == '>' && hash[sayt[0]] > sayt[len
- 1] - '0') return false;
22         if (len == 4 && sayt[1] == '>' && hash[sayt[0]] >= sayt[len
- 1] - '0') return false;
23         if (len == 4 && sayt[1] == '<' && hash[sayt[0]] <= sayt[len
- 1] - '0') return false;
24     } else {
25         if (sayt[1] == '=' && hash[sayt[0]] != sayt[len - 1] - '0')
return false;
26         if (sayt[1] == '!' && hash[sayt[0]] == sayt[len - 1] - '0')
return false;
27         if (len == 3 && sayt[1] == '<' && hash[sayt[0]] >= sayt[len
- 1] - '0') return false;
28         if (len == 3 && sayt[1] == '>' && hash[sayt[0]] <= sayt[len
- 1] - '0') return false;
29         if (len == 4 && sayt[1] == '>' && hash[sayt[0]] < sayt[len
- 1] - '0') return false;
30         if (len == 4 && sayt[1] == '<' && hash[sayt[0]] >
sayt[say[i].length() - 1] - '0') return false;
31     }
32 }
33 cnt++;
34 return true;
35 }
36 int main() {
37     for (int i = 0; i < 5; i++)
38         cin >> say[i];
39     do {
40         if (check(ranks)) cout << ranks << endl;
41     } while (next_permutation(ranks.begin(), ranks.end()));
42     cout << cnt;
43     return 0;
44 }

```

## ADV-143. 扶老奶奶过街

一共有5个红领巾，编号分别为A、B、C、D、E，老奶奶被他们其中一个扶过了马路。

五个红领巾各自说话：

A：我和E都没有扶老奶奶

B：老奶奶是被C和E其中一个扶过大街的

C：老奶奶是被我和D其中一个扶过大街的

D：B和C都没有扶老奶奶过街

E：我没有扶老奶奶

已知五个红领巾中有且只有2个人说的是真话，请问是谁扶这老奶奶过了街？

若有多个答案，在一行中输出，编号之间用空格隔开。

例如

A B C D E（这显然不是正确答案）

分析：建立一个含有5个元素的数组，分别代表abcde五个人。逐个假设abcde是扶老奶奶过街的人，判断他们说话为真的个数是否为2，为2的时候输出

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      for(int i = 0; i < 5; i++) {
5          int a[5] = {0};
6          a[i] = 1;
7          int sum = 0;
8          if(a[0] == 0 && a[4] == 0)
9              sum++;
10         if(a[2] == 1 || a[4] == 1)
11             sum++;
12         if(a[2] == 1 || a[3] == 1)
13             sum++;
14         if(a[1] == 0 && a[2] == 0)
15             sum++;
16         if(a[4] == 0)
17             sum++;
18         if(sum == 2)
19             cout << (char)('A' + i) << " ";
20     }
21     return 0;
22 }
```

## ADV-144. 01背包

问题描述

给定N个物品,每个物品有一个重量W和一个价值V.你有一个能装M重量的背包.问怎么装使得所装价值最大.每个物品只有一个.

输入格式

输入的第一行包含两个整数n, m， 分别表示物品的个数和背包能装重量。

以后N行每行两个数Wi和Vi,表示物品的重量和价值

输出格式

输出1行，包含一个整数，表示最大价值。

样例输入

3 5

2 3

3 5

4 7

样例输出

8

数据规模和约定

$1 \leq N \leq 200, M \leq 5000$ .

分析：dp[i][j]表示前i件物品选择部分装入体积为j的背包后，背包当前的最大价值，

一共有n件物品，那么dp[n][m]就是前n件物品选择部分装入容量为m的背包后，背包内物品的最大价值

1.当当前输入的物品体积大于背包容量，则不装入背包，dp[i][j] = dp[i-1][j];

2.当当前输入的物品体积小于等于背包容量，考虑装或者不装两种状态，取体积最大的那个：dp[i][j] = max(dp[i-1][j], dp[i-1][j-w] + v);

```
1  #include <iostream>
2  using namespace std;
3  int dp[201][5001];
4  int main() {
5      int n, m;
6      cin >> n >> m;
7      for(int i = 1; i <= n; i++) {
8          int w, v;
9          cin >> w >> v;
10         for(int j = 1; j <= m; j++) {
11             if (j >= w)
12                 dp[i][j] = max(dp[i-1][j], dp[i-1][j-w] + v);
13             else
14                 dp[i][j] = dp[i-1][j];
15         }
16     }
17     cout << dp[n][m];
18     return 0;
19 }
```

## ADV-145. 铺地毯

问题描述

为了准备一个学生节，组织者在会场的一片矩形区域（可看做是平面直角坐标系的第一象限）铺上一些矩形地毯。一共有n张地毯，编号从1到n。现在将这些地毯按照编号从小到大的顺序平行于坐标轴先后铺设，后铺的地毯覆盖在前面已经铺好的地毯之上。地毯铺设完成后，组织者想知道覆盖地面某个点的最上面的那张地毯的编号。注意：在矩形地毯边界和四个顶点上的点也算被地毯覆盖。

输入格式

输入共  $n+2$  行。

第一行，一个整数  $n$ ，表示总共有  $n$  张地毯。

接下来的  $n$  行中，第  $i+1$  行表示编号  $i$  的地毯的信息，包含四个正整数  $a, b, g, k$ ，每两个整数之间用一个空格隔开，分别表示铺设地毯的左下角的坐标  $(a, b)$  以及地毯在  $x$  轴和  $y$  轴方向的长度。

第  $n+2$  行包含两个正整数  $x$  和  $y$ ，表示所求的地面的点的坐标  $(x, y)$ 。

输出格式

输出共 1 行，一个整数，表示所求的地毯的编号；若此处没有被地毯覆盖则输出 -1。

样例输入

```
3
1 0 2 3
0 2 3 3
2 1 3 3
2 2
```

样例输出

```
3
```

样例输入

```
3
```

样例输出

```
-1
```

数据规模和约定

对于 30% 的数据，有  $n \leq 2$ ；

对于 50% 的数据， $0 \leq a, b, g, k \leq 100$ ；

对于 100% 的数据，有  $0 \leq n \leq 10,000$ ， $0 \leq a, b, g, k \leq 100,000$ 。

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  struct blan {
5      int a, b, x, y;
6  };
7  int main() {
8      int n, a, b, x, y, xx, yy;
9      cin >> n;
10     vector<blan> v(n);
11     for (int i = 0; i < n; i++)
12         scanf("%d %d %d %d", &v[i].a, &v[i].b, &v[i].x, &v[i].y);
13     scanf("%d %d", &xx, &yy);
14     for (int i = v.size() - 1; i >= 0; i--) {
15         if (xx >= v[i].a && xx <= v[i].a + v[i].x && yy >= v[i].b && yy
            <= v[i].b + v[i].y) {
```

```

16         cout << i + 1;
17         return 0;
18     }
19 }
20 cout << -1;
21 return 0;
22 }

```

## ADV-146. 计算器

### 【问题描述】

王小二的计算器上面的LED显示屏坏掉了，于是他找到了在计算器维修与应用系学习的你来为他修计算器。

屏幕上可以显示0~9的数字，其中每个数字由7个小二极管组成，各个数字对应的表示方式如图所示：



为了排除电路故障，现在你需要计算，将数字A变为数字B需要经过多少次变换？

注意：现在将其中每段小二极管的开和关都定义为一次变换。例如数字1变为2是5次操作。

### 【输入格式】

第一行为一个正整数L，表示数码的长度。

接下来两行是两个长度为L的数字A和B，表示要把数字A变成数字B（数字可以以0开头）。

### 【输出格式】

一行一个整数，表示这些小二极管一共要变换多少次。

### 【样例输入1】

```

3
101
025

```

### 【样例输出1】

```

12

```

### 【样例输入2】

```

8
19920513
20111211

```



## 【样例输出2】

27

## 【数据范围】

$L \leq 100$

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a[10][10] = {
5          {0, 4, 3, 3, 4, 3, 2, 3, 1, 2},
6          {4, 0, 5, 3, 2, 5, 6, 1, 5, 4},
7          {3, 5, 0, 2, 5, 4, 3, 4, 2, 3},
8          {3, 3, 2, 0, 3, 2, 3, 2, 2, 1},
9          {4, 2, 5, 3, 0, 3, 4, 3, 3, 2},
10         {3, 5, 4, 2, 3, 0, 1, 4, 2, 1},
11         {2, 6, 3, 3, 4, 1, 0, 5, 1, 2},
12         {3, 1, 4, 2, 3, 4, 5, 0, 4, 3},
13         {1, 5, 2, 2, 3, 2, 1, 4, 0, 1},
14         {2, 4, 3, 1, 2, 1, 2, 3, 1, 0}
15     };
16     int n;
17     cin >> n;
18     string s, m;
19     cin >> s >> m;
20     int cnt = 0;
21     for(int i = 0; i < n; i++) {
22         cnt += a[s[i] - '0'][m[i] - '0'];
23     }
24     cout << cnt;
25     return 0;
26 }
```

## ADV-147. 学霸的迷宫

### 问题描述

学霸抢走了大家的作业，班长为了帮同学们找回作业，决定去找学霸决斗。但学霸为了不要别人打扰，住在一个城堡里，城堡外面是一个二维的格子迷宫，要进城堡必须得先通过迷宫。因为班长还有妹子要陪，磨刀不误砍柴功，他为了节约时间，从线人那里搞到了迷宫的地图，准备提前计算最短的路线。可是他现在正向妹子解释这件事情，于是就委托你帮他找一条最短的路线。

### 输入格式

第一行两个整数n, m, 为迷宫的长宽。

接下来n行，每行m个数，数之间没有间隔，为0或1中的一个。0表示这个格子可以通过，1表示不可以。假设你现在已经在迷宫坐标(1,1)的地方，即左上角，迷宫的出口在(n,m)。每次移动时只能向上下左右4个方向移动到另外一个可以通过的格子里，每次移动算一步。数据保证(1,1)，(n,m)可以通过。

输出格式

第一行一个数为需要的最少步数K。

第二行K个字符，每个字符 $\in \{U,D,L,R\}$ ，分别表示上下左右。如果有多条长度相同的最短路径，选择在此表示方法下字典序最小的一个。

样例输入

Input Sample 1:

3 3  
001  
100  
110

Input Sample 2:

3 3  
000  
000  
000

样例输出

Output Sample 1:

4  
RDRD

Output Sample 2:

4  
DDRR

数据规模和约定

有20%的数据满足： $1 \leq n, m \leq 10$

有50%的数据满足： $1 \leq n, m \leq 50$

有100%的数据满足： $1 \leq n, m \leq 500$ 。

分析：用广度优先搜索解决。bfs。100分。如果用DFS可能会超时～

```
1 #include <iostream>
2 using namespace std;
3 struct note {
4     int x;
5     int y;
6     int f;
7     int s;
8     char dir;
9 };
```

```

10
11 int main() {
12     char a[501][501];
13     int n, m;
14     cin >> n >> m;
15     for (int i = 1; i <= n; i++) {
16         for (int j = 1; j <= m; j++) {
17             cin >> a[i][j];
18         }
19         getchar();
20     }
21     int head = 1;
22     int tail = 1;
23     int book[501][501] = {0};
24     book[1][1] = 1;
25
26     int next[4][2] = {
27         {1, 0},
28         {0, -1},
29         {0, 1},
30         {-1, 0}
31     };
32     int tx = 1;
33     int ty = 1;
34
35     struct note que[250001];
36     que[tail].x = 1;
37     que[tail].y = 1;
38     que[tail].f = 0;
39     que[tail].s = 0;
40     tail++;
41     int flag = 0;
42     while (head < tail) {
43         for (int k = 0; k <= 3; k++) {
44             tx = que[head].x + next[k][0];
45             ty = que[head].y + next[k][1];
46
47             if (tx < 1 || tx > n || ty < 1 || ty > m)
48                 continue;
49             if (a[tx][ty] == '0' && book[tx][ty] == 0) {
50                 book[tx][ty] = 1;
51                 que[tail].x = tx;
52                 que[tail].y = ty;
53                 que[tail].f = head;
54                 que[tail].s = que[head].s + 1;
55                 if (k == 0) que[tail].dir = 'D';
56                 else if (k == 1) que[tail].dir = 'L';
57                 else if (k == 2) que[tail].dir = 'R';
58                 else if (k == 3) que[tail].dir = 'U';

```

```

59         tail++;
60     }
61     if (tx == n && ty == m) {
62         flag = 1;
63         break;
64     }
65 }
66 if (flag == 1) {
67     break;
68 }
69 head++;
70 }
71 cout << que[tail - 1].s << endl;
72 char t[250001];
73 int temp = tail - 1;
74 for (int i = que[tail - 1].s; i >= 1; i--) {
75     t[i] = que[temp].dir;
76     temp = que[temp].f;
77 }
78 for (int i = 1; i <= que[tail - 1].s; i++) {
79     cout << t[i];
80 }
81 return 0;
82 }

```

## ADV-148. 排队打水问题（贪心）

### 问题描述

有 $n$ 个人排队到 $r$ 个水龙头去打水，他们装满水桶的时间 $t_1, t_2, \dots, t_n$ 为整数且各不相同，应如何安排他们的打水顺序才能使他们总共花费的时间最少？

### 输入格式

第一行 $n, r$  ( $n \leq 500, r \leq 75$ )

第二行为 $n$ 个人打水所用的时间 $T_i$  ( $T_i \leq 100$ );

### 输出格式

最少的花费时间

### 样例输入

3 2

1 2 3

### 样例输出

7

### 数据规模和约定

其中80%的数据保证 $n \leq 10$

分析：按照时间从大到小的顺序排序后即为打水的顺序~每个人依次进入队列1...r、1...r、1...r.....对于前面可以排满的且不是队列最后一排的人，每排满一行，总时间等于自身打水的时间加上前面所有人打水的时间~对于每个队列的最后一个人(或者倒数第二个人)，总时间等于自身打水的时间加上前面这一列所有人打水的时间~

```
1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  using namespace std;
5  int main() {
6      int n, r;
7      scanf("%d %d", &n, &r);
8      int *a = new int[n];
9      for(int i = 0; i < n; i++) {
10         scanf("%d", &a[i]);
11     }
12     sort(a, a+n);
13     int ans = 0;
14     int cnt = n/r;
15     int index = 0;
16     while(cnt--) {
17         for(int j = 0; j < index; j++)
18             ans += a[j];
19         for(int i = 0; i < r; i++)
20             ans += a[index++];
21     }
22     while(index < n) {
23         int t = index % r;
24         while(t < n) {
25             ans += a[t];
26             t += r;
27         }
28         index++;
29     }
30     cout << ans;
31     delete [] a;
32     return 0;
33 }
```

## ADV-149. 特殊的质数肋骨

### 问题描述

农民约翰母牛总是产生最好的肋骨。你能通过农民约翰和美国农业部标记在每根肋骨上的数字认出它们。农民约翰确定他卖给买方的是真正的质数肋骨，是因为从右边开始切下肋骨，每次还剩下的肋骨上的数字都组成一个质数。

例如有四根肋骨的数字分别是：7 3 3 1，那么全部肋骨上的数字 7331是质数；三根肋骨 733是质

数；二根肋骨 73 是质数；当然,最后一根肋骨 7 也是质数。7331 被叫做长度 4 的特殊质数。  
写一个程序对给定的肋骨的数目  $N$  ( $1 \leq N \leq 8$ ),求出所有的特殊质数。数字1不被看作一个质数。

输入格式

单独的一行包含N。

输出格式

按顺序输出长度为 N 的特殊质数,每行一个。

样例输入

4

样例输出

2333

2339

2393

2399

2939

3119

3137

3733

3739

3793

3797

5939

7193

7331

7333

7393

分析：从第一位数字开始搜索，如果发现数字是素数，就继续搜索，否则在此处剪枝～

```
1  #include <algorithm>
2  #include <map>
3  #include <cmath>
4  #include <iostream>
5  #include <vector>
6  using namespace std;
7  int n;
8  bool is(int x){
9      if(x == 1 || x == 0) return false;
10     for(int i = 2; i*i <= x; i++)
11         if(x % i == 0) return false;
12     return true;
13 }
14 void dfs(int num, int level){
15     if(level == n){
16         cout << num << endl;
```

```

17     }else{
18         for(int i = 0; i <= 9; i++){
19             if(is(num*10+i)){
20                 dfs(num * 10 + i, level+1);
21             }
22         }
23     }
24 }
25 int main() {
26     cin >> n;
27     dfs(0,0);
28     return 0;
29 }

```

## ADV-150. 周期字串

### 问题描述

右右喜欢听故事，但是右右的妈妈总是讲一些“从前有座山，山里有座庙，庙里有个老和尚给小和尚讲故事，讲的什么呢？从前有座山……”这样循环的故事来搪塞右右。

我们定义，如果一个字符串是以一个或者一个以上的长度为k的重复字符串所连接成的，那么这个字符串就叫做周期为k的串。

例如：

字符串'abcbabcbabc'周期为3，因为它是由4个循环'abc'组成的。它同样是以6为周期（两个重复的'abcbab'）和以12为周期（一个循环'abcbabcbabc'）。

右右现在想给他的朋友大灰狼转述妈妈讲的故事，请帮他写一个程序，可以测定一个字符串的最小周期。

### 输入格式

一个最大长度为100的无空格的字符串。

### 输出格式

一个整数，表示输入的字符串的最小周期。

### 样例输入

HaHaHa

### 样例输出

2

### 样例输入

Return0

### 样例输出

7

分析：从长度为1一直到长度为len/2判断是否满足周期~不满足的话就说明周期是字符串本身的长度  
~~

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string s;
6      int len = s.length();
7      for(int i = 1; i <= len / 2; i++) {
8          int flag = 0;
9          if(len % i != 0)
10             continue;
11         string t1 = s.substr(0, i);
12         string t2;
13         for(int j = i; j < len; j = j + i) {
14             t2 = s.substr(j, i);
15             if(t1 != t2) {
16                 flag = 1;
17                 break;
18             }
19         }
20         if(flag == 0) {
21             cout << i;
22             return 0;
23         }
24     }
25     cout << len;
26     return 0;
27 }
```

## ADV-154. 质数的后代

### 问题描述

在上一季里，曾提到过质数的孤独，其实从另一个角度看，无情隔膜它们的合数全是质数的后代，因为合数可以由质数相乘结合而得。

如果一个合数由两个质数相乘而得，那么我们就叫它是质数们的直接后代。现在，给你一系列自然数，判断它们是否是质数的直接后代。

### 输入格式

第一行一个正整数T，表示需要判断的自然数数量  
接下来T行，每行一个要判断的自然数

### 输出格式

共T行，依次对于输入中给出的自然数，判断是否为质数的直接后代，是则输出Yes，否则输出No



样例输入

4  
3  
4  
6  
12

样例输出

No  
Yes  
Yes  
No

数据规模和约定

$1 \leq T \leq 20$

$2 \leq \text{要判断的自然数} \leq 105$

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int a[100005] = {0};
5  vector<int> p;
6  void getprime() {
7      a[0] = a[1] = 1;
8      for (int i = 2; i * i <= 100000; i++) {
9          if (a[i] == 0) {
10             for (int j = i * i; j <= 100000; j = j + i)
11                 a[j] = 1;
12         }
13     }
14     for (int i = 2; i <= 100000; i++)
15         if (a[i] == 0) p.push_back(i);
16 }
17 void check(int n) {
18     for (int i = 0; i < p.size(); i++) {
19         if (n % p[i] == 0 && a[n / p[i]] == 0) {
20             cout << "Yes\n";
21             return;
22         }
23     }
24     cout << "No\n";
25 }
26 int main() {
27     getprime();
28     int n, k;
29     cin >> k;
30     while (k--) {
31         cin >> n;
```

```
32     check(n);
33 }
34 return 0;
35 }
```

## ADV-155. 上帝造题五分钟

### 问题描述

第一分钟，上帝说：要有题。于是就有了L, Y, M, C

第二分钟，LYC说：要有向量。于是就有了长度为n写满随机整数的向量

第三分钟，YUHCH说：要有查询。于是就有了Q个查询，查询向量的一段区间内元素的最小值

第四分钟，MZC说：要有限。于是就有了数据范围

第五分钟，CS说：要有做题的。说完众神一哄而散，留你来收拾此题

### 输入格式

第一行两个正整数n和Q，表示向量长度和查询个数

接下来一行n个整数，依次对应向量中元素：a[0], a[1], ..., a[n-1]

接下来Q行，每行两个正整数lo, hi，表示查询区间[lo, hi]中的最小值，即 $\min(a[lo], a[lo+1], \dots, a[hi])$ 。

### 输出格式

共Q行，依次对应每个查询的结果，即向量在对应查询区间中的最小值。

### 样例输入

```
7 4
1 -1 -4 8 1 2 -7
0 0
1 3
4 5
0 6
```

### 样例输出

```
1
-4
1
-7
```

### 样例说明

第一个查询[0,0]表示求 $\min\{a[0]\}=\min\{1\}=1$

第二个查询[1,3]表示求 $\min\{a[1],a[2],a[3]\}=\min\{-1,-4,8\}=-4$

第三个查询[4,5]表示求 $\min\{a[4],a[5]\}=\min\{1,2\}=1$

第四个查询[0,6]表示查询整个向量，求 $\min\{a[0..6]\}=\min\{1,-1,-4,8,1,2,-7\}=-7$

数据规模和约定

$1 \leq n \leq 1984$ ,  $1 \leq Q \leq 1988$ ，向量中随机整数的绝对值不超过1,000

```
1  #include <iostream>
2  #include <algorithm>
3  int cmp(int a, int b) {return a < b;}
4
5  using namespace std;
6  int main() {
7      int n, q;
8      cin >> n >> q;
9      int *a = new int [n];
10     for (int i = 0; i < n; i++)
11         cin >> a[i];
12     for (int i = 0; i < q; i++) {
13         int low, high;
14         cin >> low >> high;
15         int *b = new int [high - low + 1];
16         int temp = low;
17         for (int j = 0; j < high - low + 1; j++)
18             b[j] = a[temp++];
19         sort(b, b + high - low + 1, cmp);
20         cout << b[0] << endl;
21         delete [] b;
22     }
23     delete [] a;
24     return 0;
25 }
```

## ADV-156. 分分钟的碎碎念(动态规划)

问题描述

以前有个孩子，他分分钟都在碎碎念。不过，他的念头之间是有因果关系的。他会在本子里记录每一个念头，并用箭头画出这个念头的来源于之前的哪一个念头。翻开这个本子，你一定会被互相穿梭的箭头给搅晕，现在他希望你用程序计算出这些念头中最长的一条因果链。

将念头从1到n编号，念头i来源于念头from[i]，保证from[i]<i，from[i]=0表示该念头没有来源念头，只是脑袋一抽，灵光一现。

输入格式

第一行一个正整数n表示念头的数量

接下来n行依次给出from[1], from[2], ..., from[n]

## 输出格式

共一行，一个正整数L表示最长的念头因果链中的念头数量

## 样例输入

8  
0  
1  
0  
3  
2  
4  
2  
4

## 样例输出

3

## 样例说明

最长的因果链有：

1->2->5 (from[5]=2,from[2]=1,from[1]=0)

1->2->7 (from[7]=2,from[2]=1,from[1]=0)

3->4->6 (from[6]=4,from[4]=3,from[3]=0)

3->4->8 (from[8]=4,from[4]=3,from[3]=0)

## 数据规模和约定

$1 \leq n \leq 1000$

分析：建立一个与from数组等长的数组dp，dp[i]表示当前序号能满足构成的最长的长度，dp[i]的值可以由dp[from[i]]+1得到~

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main() {
5      int n;
6      scanf("%d", &n);
7      int *dp = new int[n+1];
8      int *from = new int[n+1];
9      for(int i = 1; i <= n; i++) {
10         scanf("%d", &from[i]);
11     }
12     dp[0] = 0;
13     int maxvalue = 0;
14     for(int i = 1; i <= n; i++) {
15         dp[i] = dp[from[i]] + 1;
```

```

16         maxvalue = max(maxvalue, dp[i]);
17     }
18     cout << maxvalue;
19     delete [] dp;
20     delete [] a;
21 }

```

## ADV-157. 现代诗如蚯蚓

### 问题描述

现代诗如蚯蚓

断成好几截都不会死

字符串断成好几截

有可能完全一样

请编写程序

输入字符串

输出该字符串最多能断成多少截完全一样的子串

### 输入格式

一行，一个字符串

### 输出格式

一行，一个正整数表示该字符串最多能断成的截数

### 样例输入

abcbcabcbabc

### 样例输出

4

### 样例说明

最多能断成四个“abc”，也就是abc重复四遍便是原串

同时也能断成两个“abcbcb”

最坏情况是断成一个原串“abcbcabcbabc”

### 数据规模和约定

字符串长度 $\leq 1000$

分析：字串从长度为1试探到长度为len/2，判断该字串是否是重复的字串。保留长度最短的那个值，就可以求得最多能断开的个数~~~

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string s;
6      cin >> s;
7      int len = s.length();
8      int ans = len;
9      for(int i = 1; i < len/2; i++) {
10         if(len % i != 0)
11             continue;
12         int flag = 0;
13         string t1, t2;
14         t1 = s.substr(0, i);
15         for(int j = i; j < len; j = j + i) {
16             t2 = s.substr(j, i);
17             if(t2 != t1) {
18                 flag = 1;
19                 break;
20             }
21             t1 = t2;
22         }
23         if(flag == 0)
24             ans = ans < i ? ans : i;
25     }
26     cout << len / ans;
27     return 0;
28 }

```

## ADV-158. 新建Microsoft Word文档

### 问题描述

L正在出题，新建了一个word文档，想不好取什么名字，身旁一人惊问：“你出的题目叫《新建Microsoft Word文档》吗？”，L大喜，一拍桌子，说：“好，就叫这个名字了。”

仔细观察，当你新建一个word文档时，会得到一个名为“新建 Microsoft Word 文档.doc”的文件，再新建一个，则名为“新建 Microsoft Word 文档(2).doc”，再新建，便是“新建 Microsoft Word 文档(3).doc”。不断新建，编号不断递增。倘若你现在新建了三个文档，然后删除了“新建 Microsoft Word 文档(2).doc”，再新建就又会得到一个“新建 Microsoft Word 文档(2).doc”。

严格说，Windows在每次新建文档时，都会选取一个与已有文件编号不重复的最小正整数作为新文档的编号。

请编程模拟以上过程，支持以下两种操作

New：新建一个word文档，反馈新建的文档的编号

Delete id：删除一个编号为id的word文档，反馈删除是否成功

初始时一个文件都没有，“新建 Microsoft Word 文档.doc”的编号算作1。

## 输入格式

第一行一个正整数n表示操作次数，接下来n行，每行表示一个操作。若该行为“New”，则表示新建，为“Delete id”则表示要删除编号为id的文档，其中id为一个正整数。操作按输入顺序依次进行。

## 输出格式

对于输入的每一行，输出其反馈结果。对于新建操作，输出新建的文档的编号；对于删除操作，反馈删除是否成功：如果删除的文件存在，则删除成功，输出“Successful”，否则输出“Failed”。

## 样例输入

```
12
New
New
New
Delete 2
New
Delete 4
Delete 3
Delete 1
New
New
New
Delete 4
```

## 样例输出

```
1
2
3
Successful
2
Failed
Successful
Successful
1
3
4
Successful
```

## 数据规模和约定

操作次数（即输入的行数）不超过1481

删除编号的数值不超过2012

```
1 #include <iostream>
2 #include <algorithm>
3 #include <string>
4 using namespace std;
5
```

```

6  int cmp(int a, int b) {
7      return a < b;
8  }
9
10 int delnum(string s) {
11     int len;
12     len = s.length();
13     if(len == 11) {
14         return (s[7] - '0') * 1000 + (s[8] - '0') * 100 + (s[9] - '0') *
10 + (s[10] - '0');
15     } else if(len == 10) {
16         return (s[7] - '0') * 100 + (s[8] - '0') * 10 + (s[9] - '0');
17     } else if(len == 9) {
18         return (s[7] - '0') * 10 + (s[8] - '0');
19     } else {
20         return s[7] - '0';
21     }
22 }
23
24 int main() {
25     int n;
26     cin >> n;
27     int a[1500];
28     for (int i = 0; i < 1500; i++)
29         a[i] = 2013;
30     int b[1500] = {0};
31     int count = 1;
32     int k = 0;
33     cin.get();
34     for (int i = 0; i < n; i++) {
35
36         string s;
37         getline(cin, s);
38         int flag = 0;
39         sort(a, a+1500, cmp);
40         if (s[0] == 'N') {
41             if (a[0] == 2013) {
42                 cout << count << endl;
43                 b[k++] = count;
44                 count++;
45             } else {
46                 cout << a[0] << endl;
47                 b[k++] = a[0];
48                 a[0] = 2013;
49             }
50         } else {
51             for (int l = 0; l <= k; l++) {
52                 if(b[l] == delnum(s)) {
53                     for(int j = 0; j < 1500; j++) {

```



```

54         if (a[j] == 2013) {
55             a[j] = delnum(s);
56             cout << "Successful" << endl;
57             b[1] = 0;
58             break;
59         }
60     }
61     flag = 1;
62     break;
63 }
64 }
65 if(flag == 0) {
66     cout << "Failed" << endl;
67 }
68 }
69 }
70 return 0;
71 }

```

## ADV-162. 题目1 最大最小值

### 问题描述

给定 N 个整数，请你找出这 N 个数中最大的那个和最小的那个。

### 输入格式

第一行包含一个正整数 N 。( $1 \leq N \leq 10000$ )。

第二行为 N 个用空格隔开的整数,每个数的绝对值不超过 1000000。

### 输出格式

输出仅一行,包含两个整数 x,y, x 表示 N 个数中的最大值, y 表示 N 个数中的最小值。x,y 之间用一个空格隔开。

### 样例输入

4

2 0 1 2

### 样例输出

2 0

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  int cmp(int a, int b) { return a < b; }
6
7  int main() {

```

```

8     int n;
9     cin >> n;
10    int *a = new int [n];
11    for (int i = 0; i < n; i++)
12        cin >> a[i];
13    sort(a, a+n, cmp);
14    cout << a[n - 1] << " " << a[0];
15    delete [] a;
16    return 0;
17 }

```

## ADV-165. 超级玛丽（动态规划、递推）

### 问题描述

大家都知道“超级玛丽”是一个很善于跳跃的探险家，他的拿手好戏是跳跃，但它一次只能向前跳一步或两步。有一次，他要经过一条长为 $n$ 的羊肠小道，小道中有 $m$ 个陷阱，这些陷阱都位于整数位置，分别是 $a_1, a_2, \dots, a_m$ ，陷入其中则必死无疑。显然，如果有两个挨着的陷阱，则玛丽是无论如何也跳不过去的。

现在给出小道的长度 $n$ ，陷阱的个数及位置。求出玛丽从位置1开始，有多少种跳跃方法能到达胜利的彼岸（到达位置 $n$ ）。

### 输入格式

第一行为两个整数 $n, m$

第二行为 $m$ 个整数，表示陷阱的位置

### 输出格式

一个整数。表示玛丽跳到 $n$ 的方案数

### 样例输入

4 1

2

### 样例输出

1

### 数据规模和约定

$40 \geq n \geq 3, m \geq 1$

$n > m$ ;

陷阱不会位于1及 $n$ 上

分析：和leetcode上面那道爬楼梯问题类似，但是要注意的是，爬楼梯是爬 $n$ 的长度，而这里是从1出发到 $n$ ，只要走 $n-1$ 的长度，所以是初始化 $v[1] = 1$ ,  $v[2]$ 处如果没陷阱就是1，有陷阱就是0，然后根据状态转移方程 $v[i] = v[i-1] + v[i-2]$ ;求得 $v[n]$ 的值即为种类个数~~~

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int main() {
5      int n, m;
6      cin >> n >> m;
7      vector<int> v(n+1, -1);
8      for(int i = 0; i < m; i++) {
9          int temp;
10         cin >> temp;
11         v[temp] = 0;
12     }
13     v[1] = 1;
14     v[2] = v[2] == 0 ? v[2] : 1;
15     for(int i = 3; i <= n; i++) {
16         if(v[i] != 0)
17             v[i] = v[i-1] + v[i-2];
18     }
19     cout << v[n];
20     return 0;
21 }

```

## ADV-166. 聪明的美食家

### 问题描述

如果有人认为吃东西只需要嘴巴，那就错了。

都知道舌头有这么一个特性，“由简入奢易，由奢如简难”（据好事者考究，此规律也适合许多其他情况）。具体而言，如果是甜食，当你吃的食物不如前面刚吃过的东西甜，就很不爽了。

大宝是一个聪明的美食家，当然深谙此道。一次他来到某小吃一条街，准备从街的一头吃到另一头。为了吃得爽，他大费周章，得到了各种食物的“美味度”。他拒绝不爽的经历，不走回头路而且还要爽歪歪（爽的次数尽量多）。

### 输入格式

两行数据。

第一行Z为一个整数n，表示小吃街上小吃的数量

第二行为n个整数，分别表示n种食物的“美味度”

### 输出格式

一个整数，表示吃得爽的次数

### 样例输入

10

3 18 7 14 10 12 23 41 16 24

### 样例输出

## 数据规模和约定

美味度为0到100的整数

$n < 1000$

分析：求最长不降子序列~用动态规划解决~建立一个与序列等长的数组b~b[i]表示当前i处能够构成的最长不降子序列的长度~

所以说当前b[i]的值为前面所有数字比i处数字小的的长度的最大值+1~

最后返回整个b数组中的最大值~~

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int *a = new int [n];
7      int *b = new int [n];
8      for(int i = 0; i < n; i++)
9          cin >> a[i];
10     b[0] = 1;
11     int ans = 1;
12     for(int i = 0; i < n; i++) {
13         int maxvalue = 0;
14         for(int j = i-1; j >= 0; j--) {
15             if(a[i] >= a[j])
16                 maxvalue = max(maxvalue, b[j]);
17         }
18         b[i] = maxvalue + 1;
19         ans = max(ans, b[i]);
20     }
21     cout << ans;
22     return 0;
23 }
```

## ADV-167. 快乐司机（贪心算法）

## 问题描述

“嘟嘟嘟嘟嘟嘟

喇叭响

我是汽车小司机

我是小司机

我为祖国运输忙

运输忙”

这是儿歌“快乐的小司机”。话说现在当司机光有红心不行，还要多拉快跑。多拉不是超载，是要让所载货物价值最大，特别是在当前油价日新月异的时候。司机所拉货物为散货，如大米、面粉、沙石、泥土.....

现在知道了汽车核载重量为 $w$ ，可供选择的物品的数量 $n$ 。每个物品的重量为 $g_i$ ，价值为 $p_i$ 。求汽车可装载的最大价值。（ $n < 10000, w < 10000, 0 < g_i \leq 100, 0 \leq p_i \leq 100$ ）

输入格式

输入第一行为由空格分开的两个整数 $n$   $w$

第二行到第 $n+1$ 行，每行有两个整数，由空格分开，分别表示 $g_i$ 和 $p_i$

输出格式

最大价值（保留一位小数）

样例输入

```
5 36
99 87
68 36
79 43
75 94
7 35
```

样例输出

```
71.3
```

解释：

先装第5号物品，得价值35，占用重量7

再装第4号物品，得价值36.346,占用重量29

最后保留一位小数，得71.3

分析：贪心算法，因为可以装取一部分的货物，所以每次选择(价值/数量)最大的那个货物即可~~先按照除得的单价从大到小排序，然后按照能装载的最大限度从左到右依次装载货物，直到装不下为止~~

```
1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  using namespace std;
5  struct node {
6      int a;
7      int b;
8      double c;
9  };
10 int cmp1(node t1, node t2) {
11     return t1.c > t2.c;
12 }
13 int main() {
```

```

14     int n, w;
15     cin >> n >> w;
16     node *arr = new node [n];
17     for(int i = 0; i < n; i++) {
18         cin >> arr[i].a >> arr[i].b;
19         arr[i].c = arr[i].b * 1.0 / arr[i].a;
20     }
21     sort(arr, arr+n, cmp1);
22     double ans = 0.0;
23     int j = 0;
24     while(w > 0) {
25         if(arr[j].a < w) {
26             w = w - arr[j].a;
27             ans = ans + arr[j].b;
28         } else {
29             ans = ans + w * 1.0 / arr[j].a * arr[j].b;
30             w = 0;
31         }
32         j++;
33     }
34     printf("%.1f", ans);
35     return 0;
36 }

```

## ADV-169. 士兵排队问题

### 试题

有N个士兵( $1 \leq N \leq 26$ ), 编号依次为A,B,C,..., 队列训练时, 指挥官要把一些士兵从高到矮一次排成一行, 但现在指挥官不能直接获得每个人的身高信息, 只能获得“P1比P2高”这样的比较结果( $P1、P2 \in A,B,C,...,Z$ , 记为  $P1 > P2$ ), 如“A>B”表示A比B高。

请编一程序, 根据所得到的比较结果求出一种符合条件的排队方案。

(注: 比较结果中没有涉及的士兵不参加排队)

### 输入要求

比较结果从文本文件中读入 (文件由键盘输入), 每个比较结果在文本文件中占一行。

### 输出要求

若输入数据无解, 打印“No Answer!”信息, 否则从高到矮一次输出每一个士兵的编号, 中间无分割符, 并把结果写入文本文件中, 文件由键盘输入:

### 样例输入

A>B  
B>D  
F>D

### 样例输出

AFBD

分析：高>矮，记作矮入度+1。每次找到没有排过队并且入度为0的士兵，拉出来排队，然后把紧跟后面的士兵的入度-1，一直这样循环。当找不到没排队且入度为0的士兵，判断：如果士兵全部排好了，正常输出；否则出现了环，无法确定顺序，输出No Answer!

```
1  #include <iostream>
2  #include <cstring>
3  #include <vector>
4  #include <set>
5  using namespace std;
6  int main() {
7      int in[151], vis[150] = {0};
8      set<int> se;
9      vector<int> a[150];
10     memset(in, -1, sizeof(in));
11     string s, ans;
12     while (cin >> s) {
13         if (in[s[0]] == -1) in[s[0]] = 0;
14         if (in[s[2]] == -1) in[s[2]] = 0;
15         in[s[2]]++;
16         a[s[0]].push_back(s[2]);
17         se.insert(s[0]);
18         se.insert(s[2]);
19     }
20     while (1) {
21         int cnt = 0, begin;
22         for (int i = 1; i <= 150; i++) {
23             if (in[i] == 0 && vis[i] == 0) {
24                 begin = i;
25                 cnt++;
26             }
27         }
28         if (cnt == 0) {
29             if (ans.length() == se.size()) {
30                 cout << ans;
31             } else
32                 cout << "No Answer!";
33             break;
34         } else {
35             ans += (char) begin;
36             vis[begin] = 1;
37             for (int i = 0; i < a[begin].size(); i++)
38                 in[a[begin][i]]--;
39         }
40     }
41 }
42 return 0;
43 }
```

## ADV-170. 数字黑洞

## 问题描述

任意一个四位数，只要它们各个位上的数字是不全相同的，就有这样的规律：

- 1)将组成该四位数的四个数字由大到小排列，形成由这四个数字构成的最大的四位数；
- 2)将组成该四位数的四个数字由小到大排列，形成由这四个数字构成的最小的四位数(如果四个数中含有0，则得到的数不足四位)；
- 3)求两个数的差，得到一个新的四位数(高位零保留)。

重复以上过程，最后一定会得到的结果是6174。

比如：4312 3087 8352 6174，经过三次变换，得到6174

## 输入格式

一个四位整数，输入保证四位数字不全相同

## 输出格式

一个整数，表示这个数字经过多少次变换能得到6174

## 样例输入

4312

## 样例输出

3

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int cmp1(int a, int b) {return a > b;}
5  int cmp2(int a, int b) {return a < b;}
6  int main() {
7      int a[4];
8      string s;
9      cin >> s;
10     for(int i = 0; i < 4; i++) {
11         a[i] = s[i] - '0';
12     }
13     int res = a[0] * 1000 + a[1] * 100 + a[2] * 10 + a[3];
14     int cnt = 0;
15     while(res != 6174) {
16         cnt++;
17         sort(a, a+4, cmp1);
18         int big = a[0] * 1000 + a[1] * 100 + a[2] * 10 + a[3];
19         sort(a, a+4, cmp2);
20         int small = a[0] * 1000 + a[1] * 100 + a[2] * 10 + a[3];
21         res = big - small;
22         a[0] = res / 1000;
23         a[1] = res / 100 - res / 1000 * 10;
```



```

24         a[2] = res / 10 - res / 100 * 10;
25         a[3] = res % 10;
26     }
27     cout << cnt;
28     return 0;
29 }

```

## ADV-171. 身份证号码升级

### 问题描述

从1999年10月1日开始，公民身份证号码由15位数字增至18位。(18位身份证号码简介)。升级方法为：

- 1、把15位身份证号码中的年份由2位(7,8位)改为四位。
- 2、最后添加一位验证码。验证码的计算方案：

将前 17 位分别乘以对应系数 (7 9 10 5 8 4 2 1 6 3 7 9 10 5 8 4 2) 并相加，然后除以 11 取余数，0-10 分别对应 1 0 × 9 8 7 6 5 4 3 2。

请编写一个程序，用户输入15位身份证号码，程序生成18位身份证号码。假设所有要升级的身份证的四位年份都是19××年

### 输入格式

一个15位的数字串，作为身份证号码

### 输出格式

一个18位的字符串，作为升级后的身份证号码

### 样例输入

110105491231002

### 样例输出

11010519491231002x

### 数据规模和约定

不用判断输入的15位字符串是否合理

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string a;
6      cin >> a;
7      int result = 0;
8      int b[15] = {7, 9, 10, 5, 8, 4, 6, 3, 7, 9, 10, 5, 8, 4, 2};
9      for (int i = 0; i < 15; i++)
10         result += (a[i] - '0') * b[i];
11     result = result % 11;

```

```

12     string c = "10x98765432";
13     for (int i = 0; i <= 5; i++)
14         cout << a[i];
15     cout << "19";
16     for (int i = 6; i <= 14; i++)
17         cout << a[i];
18     cout << c[result];
19     return 0;
20 }

```

## ADV-173. 淘淘的名单

### 问题描述

by ZBY... :) 淘淘拿到了一份名单，他想对上面的名字进行处理，挑出一些特殊的名字，他请你来帮忙。

淘淘关注以下名字：

如果这个名字是“WYS”，他希望你的程序输出“KXZSMR”。

如果这个名字是“CQ”，他希望你的程序输出“CHAIQIANG”。

如果这个名字是“LC”，他希望你的程序输出“DRAGONNET”。

如果这个名字是“SYT”或“SSD”或“LSS”或“LYF”，他希望你的程序输出“STUDYFATHER”。

如果这个名字与上述任意名字都不相同，他希望你的程序输出“DENOMINATOR”。

### 输入格式

第一行有一个整数N，表示淘淘手中名单里的人数。

接下来N行，每行有一个字符串，即名单里的人名。

### 输出格式

输出N行，每行输出每个人名的判断结果。

### 样例输入

```

9
WYS
CQ
WYS
LC
SYT
SSD
LSS
LYF
ZBY

```

### 样例输出

KXZSMR  
CHAIQIANG  
KXZSMR  
DRAGONNET  
STUDYFATHER  
STUDYFATHER  
STUDYFATHER  
STUDYFATHER  
DENOMINATOR

数据规模和约定

对于 50% 数据， $N \leq 1000$ ，且名单中的名字仅可能为“WYS”、“CQ”、“LC”三者之一，没有其他的名字。

对于 100% 数据， $N \leq 10000$ ，人名仅由大写字母组成，长度不超过5。

分析：水题。。但是原谅我才知道switch里面只能char型或者int型。。我还以为可以string类型。。。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      string s;
7      for(int i = 0; i < n; i++) {
8          cin >> s;
9          if(s == "WYS") {
10             cout << "KXZSMR" << endl;
11         } else if(s == "CQ") {
12             cout << "CHAIQIANG" << endl;
13         } else if(s == "LC") {
14             cout << "DRAGONNET" << endl;
15         } else if(s == "SSD" || s == "LSS" || s == "LYF" || s == "SYT")
16         {
17             cout << "STUDYFATHER" << endl;
18         } else {
19             cout << "DENOMINATOR" << endl;
20         }
21     }
22     return 0;
23 }
```

## ADV-175. 三个整数的排序

问题描述

输入三个数，比较其大小，并从大到小输出。

输入格式

一行三个整数。

输出格式

一行三个整数，从大到小排序。

样例输入

33 88 77

样例输出

88 77 33

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b, c, t;
5      cin >> a >> b >> c;
6      if (a < b)
7          swap(a, b);
8      if (b < c)
9          swap(b, c);
10     if (a < b)
11         swap(a, b);
12     cout << a << " " << b << " " << c;
13     return 0;
14 }
```

## ADV-176. 陶陶摘苹果

问题描述

陶陶家的院子里有一棵苹果树，每到秋天树上就会结出 $n$ 个苹果。苹果成熟的时候，陶陶就会跑去摘苹果。陶陶有个30厘米高的板凳，当她不能直接用手摘到苹果的时候，就会踩到板凳上再试试。

现在已知 $n$ 个苹果到地面的高度，以及陶陶把手伸直的时候能够达到的最大高度，请帮陶陶算一下她能够摘到的苹果的数目。假设她碰到苹果，苹果就会掉下来。

输入格式

输入包括两行数据。第一行只包括两个正整数 $n$ ( $5 \leq n \leq 200$ )和 $m$ ( $100 \leq m \leq 150$ ),表示苹果数目和桃桃伸手可达到的高度（以厘米为单位）。第二行包含 $n$ 个100到200之间（包括100和200）的整数（以厘米为单位）分别表示苹果到地面的高度，两个相邻的整数之间用一个空格隔开。

输出格式

输出包括一行，这一行只包含一个整数，表示陶陶能够摘到的苹果的数目。

样例输入

10 110

100 200 150 140 129 134 167 198 200 111

样例输出

5

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, m;
5      cin >> n >> m;
6      int cnt = 0;
7      for(int i = 0; i < n; i++) {
8          int temp;
9          cin >> temp;
10         if(m + 30 >= temp)
11             cnt++;
12     }
13     cout << cnt;
14     return 0;
15 }
```

## ADV-177. 理财计划

问题描述

银行近期推出了一款新的理财计划“重复计息储蓄”。储户只需在每个月月初存入固定金额的现金，银行就会在每个月月底根据储户账户内的金额算出该月的利息并将利息存入用户账号。现在如果某人每月存入 $k$ 元，请你帮他计算一下， $n$ 月后，他可以获得多少收益。

输入格式

输入数据仅一行，包括两个整数 $k$  ( $100 \leq k \leq 10000$ )、 $n$  ( $1 \leq n \leq 48$ )和一个小数 $p$  ( $0.001 \leq p \leq 0.01$ )，分别表示每月存入的金额、存款时长、存款利息。

输出格式

输出数据仅一个数，表示可以得到的收益。

样例输入

1000 6 0.01

样例输出

213.53

分析：不知道给出的样例是不是有问题。。我用xcode运行出来结果是213.54

结果提交了还AC了。。。

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main() {
```

```

5     int k, n;
6     double p;
7     cin >> k >> n >> p;
8     double ans = 0;
9     for(int i = 0; i < n; i++) {
10         ans = ans + k;
11         ans = ans * (1 + p);
12     }
13     printf("%.2f", ans - n * k);
14     return 0;
15 }

```

## ADV-178. 简单加法

### 问题描述

小于10的自然数中有四个数字能除尽3或5（3， 5， 6， 9），它们的和为23。

请计算所有小于1000的自然数中能除尽3或5的数字的合。然后使用标准输出cout，输出你的结果。

### 输入格式

无。

### 输出格式

一行一个整数，表示你的结果。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int result = 0;
5      for (int i = 1; i < 1000; i++) {
6          if (i % 3 == 0 || i % 5 == 0)
7              result += i;
8      }
9      cout << result;
10     return 0;
11 }

```

## ADV-179. 解二元一次方程组

### 问题描述

给定一个二元一次方程组，形如：

$$a * x + b * y = c;$$

$$d * x + e * y = f;$$

x,y代表未知数，a, b, c, d, e, f为参数。

求解x,y

输入格式

输入包含六个整数: a, b, c, d, e, f;

输出格式

输出为方程组的解, 两个整数x, y。

样例输入

例:

3 7 41 2 1 9

样例输出

例:

2 5

数据规模和约定

$0 \leq a, b, c, d, e, f \leq 2147483647$

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int a, b, c, d, e, f;
5     cin >> a >> b >> c >> d >> e >> f;
6     long long int x, y;
7     y = (c * d - f * a) / (b * d - e * a);
8     x = (c - b * y) / a;
9     cout << x << " " << y;
10    return 0;
11 }
```

## ADV-180. 陶陶摘苹果2

问题描述

陶陶家的院子里有一棵苹果树, 每到秋天树上就会结出n个苹果。苹果成熟的时候, 陶陶就会跑去摘苹果。陶陶有个30厘米高的板凳, 当她不能直接用手摘到苹果的时候, 就会踩到板凳上再试试。

现在已知n个苹果到地面的高度, 以及陶陶把手伸直的时候能够达到的最大高度。假设她碰到苹果, 苹果就会掉下来。请帮陶陶算一下, 经过她的洗劫后, 苹果树上还有几个苹果。

输入格式

输入包括两行数据。第一行只包括两个正整数n( $5 \leq n \leq 200$ )和m( $60 \leq m \leq 200$ ), 表示苹果数目和桃桃伸手可达到的高度 (以厘米为单位)。第二行包含n个100到200之间 (包括100和200) 的整数 (以厘米为单位) 分别表示苹果到地面的高度, 两个相邻的整数之间用一个空格隔开。

输出格式

输出包括一行，这一行只包含一个整数，表示陶陶不能够摘到的苹果的数目。

样例输入

10 110

100 200 150 140 129 134 167 198 200 111

样例输出

5

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n, m;
5      cin >> n >> m;
6      int cnt = n;
7      for(int i = 0; i < n; i++) {
8          int temp;
9          cin >> temp;
10         if(m + 30 >= temp)
11             cnt--;
12     }
13     cout << cnt;
14     return 0;
15 }
```

## ADV-181. 质因数2

将一个正整数 $N(1 < N < 32768)$ 分解质因数，把质因数按从小到大的顺序输出。最后输出质因数的个数。

输入格式

一行，一个正整数

输出格式

两行，第一行为用空格分开的质因数

第二行为质因数的个数

样例输入

66

样例输出

2 3 11

3

样例输入

90



样例输出

2 3 3 5

4

样例输入

37

样例输出

37

1

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n;
5      cin >> n;
6      int cnt = 0;
7      while(n != 1) {
8          for(int i = 2; i <= n; i++) {
9              if(n % i == 0) {
10                 cout << i << " ";
11                 cnt++;
12                 n = n / i;
13                 break;
14             }
15         }
16     }
17     cout << endl << cnt;
18     return 0;
19 }
```

## ADV-182. 前10名

问题描述

数据很多，但我们经常只取前几名，比如奥运只取前3名。现在我们有n个数据，请按从大到小的顺序，输出前10个名数据。

输入格式

两行。

第一行一个整数n，表示要对多少个数据

第二行有n个整数，中间用空格分隔。表示n个数据。

输出格式

一行，按从大到小排列的前10个数据，每个数据之间用一个空格隔开。

样例输入

26

54 27 87 16 63 40 40 22 61 6 57 70 0 42 11 50 13 5 56 7 8 86 56 91 68 59

样例输出

91 87 86 70 68 63 61 59 57 56

数据规模和约定

$10 \leq n \leq 200$ , 各个整数不超出整型范围

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int cmp1(int a, int b) {return a > b;}
5  int main() {
6      int n;
7      cin >> n;
8      int *a = new int[n];
9      for(int i = 0; i < n; i++) {
10         cin >> a[i];
11     }
12     sort(a, a+n, cmp1);
13     for(int i = 0; i < 10; i++)
14         cout << a[i] << " ";
15     delete [] a;
16     return 0;
17 }
```

## ADV-184. 素数求和

问题描述

输入一个自然数n, 求小于等于n的素数之和

样例输入

2

样例输出

2

数据规模和约定

测试样例保证  $2 \leq n \leq 2,000,000$

```
1  #include <iostream>
2  using namespace std;
3  int v[2000001];
4  int main() {
```

```

5     int n;
6     cin >> n;
7     for(int i = 2; i * i <= n; i++) {
8         if(v[i] == 1)
9             continue;
10        for(int j = i * i; j <= n; j = j + i)
11            v[j] = 1;
12    }
13    long long int cnt = 0;
14    for(int i = 2; i <= n ; i++) {
15        if(v[i] == 0) {
16            cnt += i;
17        }
18    }
19    cout << cnt;
20    return 0;
21 }

```

## ADV-187. 勾股数

### 问题描述

勾股数是一组三个自然数， $a < b < c$ ，以这三个数为三角形的三条边能够形成一个直角三角形

输出所有 $a + b + c \leq 1000$ 的勾股数

$a$ 小的先输出； $a$ 相同的， $b$ 小的先输出。

### 输出格式

每行为一组勾股数，用空格隔开

### 样例输出

例如，结果的前三行应当是

3 4 5

5 12 13

6 8 10

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      for(int i = 1; i <= 333; i++) {
5          for(int j = i+1; j <= 666; j++) {
6              for(int k = j+1; k <= 999; k++) {
7                  if(i + j + k <= 1000 && i*i + j*j == k*k)
8                      cout << i << " " << j << " " << k << endl;
9              }
10         }
11     }
12     return 0;
13 }

```

## ADV-188. 排列数

### 问题描述

0、1、2三个数字的全排列有六种，按照字母序排列如下：

012、021、102、120、201、210

输入一个数n

求0~9十个数字的全排列中的第n个（第1个为0123456789）。

### 输入格式

一行，包含一个整数n

### 输出格式

一行，包含一组10个数字的全排列

### 样例输入

1

### 样例输出

0123456789

### 数据规模和约定

$0 < n \leq 10!$

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int main() {
5      string s = "0123456789";
6      int n;
7      cin >> n;
8      int cnt = 1;

```

```

9      do {
10          if(cnt == n) {
11              cout << s;
12              break;
13          }
14          cnt++;
15      }while(next_permutation(s.begin(), s.end()));
16      return 0;
17  }

```

## ADV-189. 连接乘积

### 问题描述

192这个数很厉害，用它分别乘以1、2、3，会得到：

$192 \times 1 = 192$

$192 \times 2 = 384$

$192 \times 3 = 576$

把这三个乘积连起来，得到192384576，正好是一个1~9的全排列

我们把上面的运算定义为连接乘积：

$m \times (1 \dots n) = k$ （其中 $m > 0$ 且 $n > 1$ ，对于上例， $m = 192$ 、 $n = 3$ 、 $k = 192384576$ ）

即 $k$ 是把 $m$ 分别乘以1到 $n$ 的乘积连接起来得到的，则称 $k$ 为 $m$ 和 $n$ 的连接乘积。

按字典序输出所有不同的连接乘积 $k$ ，满足 $k$ 是1~9的全排列

### 输出格式

每个 $k$ 占一行

### 样例输出

显然，结果中应包含一行：

192384576

```

1  #include <iostream>
2  using namespace std;
3  int gcd(int a, int b) {
4      if (b == 0) return a;
5      return gcd(b, a % b);
6  }
7  int main() {
8      int a, b, c, ans;
9      cin >> a >> b >> c;
10     ans = a * b / gcd(a, b);
11     ans = ans * c / gcd(ans, c);
12     cout << ans;
13     return 0;
14 }

```

## ADV-193. 盾神与条状项链

## 问题描述

有一天，盾神捡到了好多好多五颜六色的珠子！他心想这些珠子这么漂亮，可以做成一条项链然后送给他心仪的女生~于是他用其中一些珠子做成了长度为 $n$ 的项链。当他准备把项链首尾相接的时候，土方进来了。

“哇这么恶心的项链你也做得出来！！！”

盾神自知审美不是他的长项，于是他很谦虚地请教土方，怎么才能把项链做得漂亮。

“这个嘛~首先你要在这里加上一个这种颜色的珠子，然后在这里去掉这个珠子，然后.....，最后你看看是不是漂亮很多咧~”土方一下子说出了 $m$ 个修改步骤。

盾神觉得这个用人工做太麻烦了，于是交给了你。

## 输入格式

第一行两个数，分别为 $n$ ， $m$ 。

第二行 $n$ 个数，表示盾神一开始的项链。第 $i$ 个数表示第 $i$ 颗珠子的颜色。

接下来 $m$ 行，为以下形式之一：

ADD P Q：表示在颜色为P的珠子前面加上一个颜色为Q的珠子。

DEL P：表示把颜色为P的珠子去掉，如果它不在端点处，则需要把它旁边的两颗珠子连起来。例如某时刻项链状态为1 4 5 8，则执行DEL 4会变成1 5 8，执行DEL 1会变成4 5 8。

输入保证在每次操作之前，项链有颜色为P的珠子，且任意时刻珠子颜色互不相同。

## 输出格式

第一行为一个数 $len$ ，为做完所有操作后，项链的长度。

第二行 $len$ 个数，表示此时项链的状态。第 $i$ 个数表示第 $i$ 颗珠子的颜色。

## 样例输入

10 5

1 2 3 4 5 6 7 8 9 10

DEL 5

ADD 7 5

DEL 10

ADD 4 20

ADD 20 12

## 样例输出

11

1 2 3 12 20 4 6 5 7 8 9

## 数据规模和约定

表示颜色的数字不超过 $10^5$ 的正数,  $1 \leq n \leq 10^4$ ,  $1 \leq m \leq 10^4$ 。

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int main() {
5      int n, m;
6      cin >> n >> m;
7      vector<int> v(n);
8      for(int i = 0; i < n; i++) {
9          cin >> v[i];
10     }
11     for(int i = 0; i < m; i++) {
12         string s;
13         cin >> s;
14         if(s == "DEL") {
15             int p;
16             cin >> p;
17             for(vector<int>::iterator it = v.begin(); it != v.end();
18 it++) {
19                 if(*it == p) {
20                     v.erase(it);
21                     break;
22                 }
23             }
24             if(s == "ADD") {
25                 int p, q;
26                 cin >> p >> q;
27                 for(vector<int>::iterator it = v.begin(); it != v.end();
28 it++) {
29                     if(*it == p) {
30                         v.insert(it, q);
31                         break;
32                     }
33                 }
34             }
35             cout << v.size() << endl;
36             for(int i = 0; i < v.size(); i++) {
37                 cout << v[i] << " ";
38             }
39             return 0;
40     }
```

## ADV-194. 盾神与积木游戏（贪心）

问题描述

最近的m天盾神都去幼儿园陪小朋友们玩去了~

每个小朋友都拿到了一些积木，他们各自需要不同数量的积木来拼一些他们想要的东西。但是有的小朋友拿得多，有的小朋友拿得少，有些小朋友需要拿到其他小朋友的积木才能完成他的大作。如果某个小朋友完成了他的作品，那么他就会把自己的作品推倒，而无私地把他的所有积木都奉献出来；但是，如果他还没有完成自己的作品，他是不会把积木让出去的哟~

盾神看到这么和谐的小朋友们感到非常开心，于是想帮助他们所有人都完成他们各自的作品。盾神现在在想，这个理想有没有可能实现呢？于是把这个问题交给了他最信赖的你。

输入格式

第一行为一个数m。

接下来有m组数据。每一组的第一行为n，表示这天有n个小朋友。接下来的n行每行两个数，分别表示他现在拥有的积木数和他一共需要的积木数。

输出格式

输出m行，如果第i天能顺利完成所有作品，输出YES，否则输出NO。

样例输入

```
2
2
2 2
1 3
3
1 5
3 3
0 4
```

样例输出

YES

NO

数据规模和约定

$1 \leq n \leq 10000$ ,  $1 \leq m \leq 10$ 。

分析：首先将已经能够完成作品的积木个数释放，然后优先分配给需要积木最少的人

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  struct node {
5      int t1, t2, res;
6  };
7  int cmp1(node p, node q) {
8      return p.res > q.res;
9  }
10 int main() {
```



```

11     int m, n;
12     cin >> m;
13     for(int i = 0; i < m; i++) {
14         cin >> n;
15         int t = 0;
16         int flag = 0;
17         node *a = new node[n];
18         for(int j = 0; j < n; j++) {
19             cin >> a[j].t1 >> a[j].t2;
20             a[j].res = a[j].t1 - a[j].t2;
21             if(a[j].res >= 0) {
22                 a[j].res = -99999999;
23                 t += a[j].t1;
24             }
25         }
26         sort(a, a+n, cmp1);
27         int k = 0;
28         while(a[k].res != -99999999) {
29             t += a[k].res;
30             if(t < 0) {
31                 cout << "NO" << endl;
32                 flag = 1;
33                 break;
34             }
35             t += a[k].t2;
36             k++;
37         }
38         if(flag == 0)
39             cout << "YES" << endl;
40     }
41     return 0;
42 }

```

## ADV-197. P1001

当两个比较大的整数相乘时，可能会出现数据溢出的情形。为避免溢出，可以采用字符串的方法来实现两个大数之间的乘法。

具体来说，首先以字符串的形式输入两个整数，每个整数的长度不会超过8位，然后把它们相乘的结果存储在另一个字符串当中

（长度不会超过16位），最后把这个字符串打印出来。例如，假设用户输入为：62773417和12345678，

则输出结果为：774980393241726。

输入：

62773417 12345678

输出：

```

1  #include <iostream>
2  #include <string>
3  #include <cmath>
4  using namespace std;
5  int main() {
6      string a, b;
7      cin >> a >> b;
8      int lena = a.length();
9      int lenb = b.length();
10     char d[20][20];
11     char c[20];
12     for (int i = 0; i <= 19; i++) {
13         for (int j = 0; j <= 19; j++) {
14             d[i][j] = '0';
15         }
16         c[i] = '0';
17     }
18
19     int temp = 0;
20     int k = 0;
21     int cou = 0;
22     int t = 0;
23     for (int i = lenb - 1; i >= 0; i--) {
24         k = 0;
25         for (int j = lena - 1; j >= 0; j--) {
26             t = ((b[i] - '0') * (a[j] - '0'));
27             temp = temp + t * pow(10, k++);
28         }
29         k = 0;
30         while(temp != 0) {
31             d[cou][k++] = (temp % 10) + '0';
32             temp = temp / 10;
33         }
34         cou++;
35     }
36
37     for (int i = 0; i <= 18; i++) {
38         for (int j = 0; j <= i; j++) {
39             c[i] = (c[i] - '0' + d[j][i - j] - '0') % 10 + '0';
40             temp = temp + (d[j][i - j] - '0');
41         }
42         temp = temp / 10;
43         c[i + 1] = temp + '0';
44     }
45
46     int flag = 0;
47     for (int i = 19; i >= 0; i--) {

```

```

48         if (c[i] != '0') {
49             flag = i;
50             break;
51         }
52     }
53     for (int i = flag; i >= 0; i--)
54         cout << c[i];
55     return 0;
56 }

```

## ADV-201. 我们的征途是星辰大海

最新的火星探测机器人curiosity被困在了一个二维迷宫里，迷宫由一个个方格组成。

共有四种方格：

‘.’代表空地，curiosity可以穿过它

‘#’代表障碍物，不可穿越，不可停留

‘S’代表curiosity的起始位置

‘T’代表curiosity的目的地

NASA将会发送一系列的命令给curiosity，格式如下：“LRUD”分别代表向左，向右，向上，向下走一步。由于地球和火星之间最近时也有55000000km！所以我们必须提前判断一系列的指令会让curiosity最终处在什么样的状态，请编程完成它。

输入格式

第一行是一个整数T，代表有几个测试样例

每个测试样例第一行是一个整数N（ $1 \leq N \leq 50$ ）代表迷宫的大小（ $N \times N$ ）。随后的N行每行由N个字符串组成，代表迷宫。接下来的一行是一个整数Q，代表有多少次询问，接下来的Q行每行是一个仅由“LRUD”四个字母的组成的字符串，字符串长度小于1000。

输出格式

对于每个询问输出单独的一行：

“I get there!”：执行给出的命令后curiosity最终到达了终点。

“I have no idea!”：执行给出的命令后curiosity未能到达终点。

“I am dizzy!”：curiosity在执行命令的过程中撞到了障碍物。

“I am out!”：代表curiosity在执行命令的过程中走出了迷宫的边界。

Sample Input

```

2
2
S.
#T
2
RD
DR
3
S.#
.#.
.T#
3

```

RL

DDD

DDRR

Sample Output

I get there!

I am dizzy!

I have no idea!

I am out!

I get there!

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4  int main() {
5      int k;
6      cin >> k;
7      while (k--) {
8          int n, m, sx, sy, ex, ey;
9          char cc[100][100];
10         memset(cc, '0', sizeof(cc));
11         cin >> n;
12         for (int i = 1; i <= n; i++) {
13             for (int j = 1; j <= n; j++) {
14                 scanf(" %c ", &cc[i][j]);
15                 if (cc[i][j] == 'S') {
16                     sx = i;
17                     sy = j;
18                 }
19                 if (cc[i][j] == 'T') {
20                     ex = i;
21                     ey = j;
22                 }
23             }
24         }
25         cin >> m;
26         while (m--) {
27             int nowx = sx, nowy = sy, flag = 0;
28             string s;
29             cin >> s;
30             for (int i = 0; i < s.length(); i++) {
31                 if (s[i] == 'R') nowy++;
32                 if (s[i] == 'L') nowy--;
33                 if (s[i] == 'U') nowx--;
34                 if (s[i] == 'D') nowx++;
35                 if (cc[nowx][nowy] == '#') {
36                     flag = 1;
37                     cout << "I am dizzy!\n";
38                     break;
39                 } else if (cc[nowx][nowy] == '0') {
```

```

40         flag = 1;
41         cout << "I am out!\n";
42         break;
43     } else if (cc[nowx][nowy] == 'T') {
44         flag = 1;
45         cout << "I get there!\n";
46         break;
47     }
48 }
49 if (flag == 0)
50     cout << "I have no idea!\n";
51 }
52 }
53 return 0;
54 }

```

## ADV-202. 最长公共子序列（动态规划）

### 问题描述

给定两个字符串，寻找这两个字符串之间的最长公共子序列。

### 输入格式

输入两行，分别包含一个字符串，仅含有小写字母。

### 输出格式

最长公共子序列的长度。

### 样例输入

abcdgh

aedfhb

### 样例输出

3

### 样例说明

最长公共子序列为a, d, h。

### 数据规模和约定

字符串长度1~1000。

分析：求最长公共子序列，用动态规划~只需建立一个长宽为两个字符串长度+1的二维数组~dp[i][j]表示String a的前i个字符构成的字符串和String b的前j个字符构成的字符串这两者得到的最长公共子序列的长度为dp[i][j]~~~所以第0行和第0列所有的数都为0~

根据递推公式：

$$C[i,j] = \begin{cases} 0 & \text{若 } i=0 \text{ 或 } j=0 \\ C[i-1,j-1]+1 & \text{若 } i,j>0, x_i = y_j \\ \max\{C[i,j-1], C[i-1,j]\} & \text{若 } i,j>0, x_i \neq y_j \end{cases}$$

按一行一行的顺序填入数~

|   |   | a | b | c | d | g | h | e |
|---|---|---|---|---|---|---|---|---|
|   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | 0 |   |   |   |   |   |   |   |
| e | 0 |   |   |   |   |   |   |   |
| d | 0 |   |   |   |   |   |   |   |
| f | 0 |   |   |   |   |   |   |   |
| h | 0 |   |   |   |   |   |   |   |
| b | 0 |   |   |   |   |   |   |   |
| e | 0 |   |   |   |   |   |   |   |

=>

|   |   | a | b | c | d | g | h | e |
|---|---|---|---|---|---|---|---|---|
|   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| e | 0 |   |   |   |   |   |   |   |
| d | 0 |   |   |   |   |   |   |   |
| f | 0 |   |   |   |   |   |   |   |
| h | 0 |   |   |   |   |   |   |   |
| b | 0 |   |   |   |   |   |   |   |
| e | 0 |   |   |   |   |   |   |   |

=>

|   |   | a | b | c | d | g | h | e |
|---|---|---|---|---|---|---|---|---|
|   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| e | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| d | 0 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| f | 0 |   |   |   |   |   |   |   |
| h | 0 |   |   |   |   |   |   |   |
| b | 0 |   |   |   |   |   |   |   |
| e | 0 |   |   |   |   |   |   |   |

=>

|   |   | a | b | c | d | g | h | e |
|---|---|---|---|---|---|---|---|---|
|   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| e | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| d | 0 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| f | 0 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| h | 0 | 1 | 1 | 1 | 2 | 2 | 3 | 3 |
| b | 0 | 1 | 2 | 2 | 2 | 2 | 3 | 3 |
| e | 0 | 1 | 2 | 2 | 2 | 2 | 3 | 4 |

最后一个格子的长度就是两个字符串的最长公共子序列的长度~~

```

1  #include <iostream>
2  using namespace std;
3  int dp[1001][1001];
4  int main() {
5      string a, b;
6      cin >> a >> b;
7      for(int i = 1; i <= a.length(); i++) {
8          for(int j = 1; j <= b.length(); j++) {
9              if(a[i-1] == b[j-1])
10                 dp[i][j] = dp[i-1][j-1] + 1;
11             else
12                 dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
13         }
14     }

```

```

15     cout << dp[a.length()][b.length()];
16     return 0;
17 }

```

## ADV-203. 8皇后·改(八皇后问题)

### 问题描述

规则同8皇后问题，但是棋盘上每格都有一个数字，要求八皇后所在格子数字之和最大。

### 输入格式

一个8\*8的棋盘。

### 输出格式

所能得到的最大数字和

### 样例输入

```

1 2 3 4 5 6 7 8
9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
48 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64

```

### 样例输出

260

### 数据规模和约定

棋盘上的数字范围0~99

```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int maxvalue = 0;
5  int pic[8][8];
6
7  bool issafe(int pos[], int row) {
8      for(int i = 0; i < row; i++) {
9          if(pos[i] == pos[row] || abs(i - row) == abs(pos[i] -
10 pos[row]))
11              return false;
12      }
13  }

```

```

12     return true;
13 }
14 void dfs(int pos[], int row) {
15     if(row == 8) {
16         int sum = 0;
17         for(int i = 0; i < 8; i++) {
18             sum += pic[i][pos[i]];
19         }
20         maxvalue = sum > maxvalue ? sum : maxvalue;
21         return ;
22     }
23     for(pos[row] = 0; pos[row] < 8; pos[row]++) {
24         if(issafe(pos, row)) {
25             dfs(pos, row + 1);
26         }
27     }
28 }
29
30 int main() {
31     int pos[8];
32     for(int i = 0; i < 8; i++) {
33         for(int j = 0; j < 8; j++) {
34             cin >> pic[i][j];
35         }
36     }
37     dfs(pos, 0);
38     cout << maxvalue;
39     return 0;
40 }

```

## ADV-204. 快速幂

### 问题描述

给定A, B, P, 求 $(A^B) \bmod P$ 。

### 输入格式

输入共一行。

第一行有三个数, N, M, P。

### 输出格式

输出共一行, 表示所求。

### 样例输入

2 5 3

### 样例输出

2



数据规模和约定

共10组数据

对100%的数据，A, B为long long范围内的非负整数，P为int内的非负整数。

```
1  #include <iostream>
2  using namespace std;
3  long long p(long long a, long long b, long long m) {
4      if (b == 0) return 1;
5      if (b % 2 == 1) return p(a, b - 1, m) * (a % m) % m;
6      long long t = p(a, b / 2, m);
7      return t * t % m;
8  }
9  int main() {
10     long long a, b, m;
11     cin >> a >> b >> m;
12     cout << p(a, b, m);
13     return 0;
14 }
```

## ADV-205. 拿糖果（动态规划）

问题描述

妈妈给小B买了N块糖！但是她不允许小B直接吃掉。

假设当前有M块糖，小B每次可以拿P块糖，其中P是M的一个不大于根号下M的质因数。这时，妈妈就会在小B拿了P块糖以后再从糖堆里拿走P块糖。然后小B就可以接着拿糖。

现在小B希望知道最多可以拿多少糖。

输入格式

一个整数N

输出格式

最多可以拿多少糖

样例输入

15

样例输出

6

数据规模和约定

$N \leq 100000$

分析：动态规划问题~~首先呢~创建一个满足不大于根号下最大值MAXN的素数表，然后对素数表里面的数逐个遍历~

构建一个dp[i]数组,表示当糖果数量为i的时候所能拿的最多的糖果数量~

对于dp[i]的值：因为小B只能每次拿不大于根号下i的质因数，遍历素数表中满足条件的素数（ $\text{prime}[j] \leq \sqrt{i}$  且  $i \% \text{prime}[j] == 0$ ），更新dp[i]的值为（ $\text{dp}[i-2*\text{prime}[j]] + \text{prime}[j]$ ）的最大值~

即： $\text{dp}[i] = \max(\text{dp}[i], \text{dp}[i-2*\text{prime}[j]] + \text{prime}[j]);$

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int prime[50000];
5  int dp[100005];
6  int book[100005];
7  int cnt = 0;
8
9  void create() {
10     int len = sqrt(100005);
11     for(int i = 2; i <= len; i++) {
12         if(book[i] == 0) {
13             prime[cnt++] = i;
14             for(int j = i * i; j <= len; j = j + i)
15                 book[j] = 1;
16         }
17     }
18 }
19
20 int main() {
21     create();
22     int n;
23     cin >> n;
24     for(int i = 1; i <= n; i++) {
25         for(int j = 0; j < cnt; j++) {
26             if(prime[j] > sqrt(i))
27                 break;
28             if(i % prime[j] == 0)
29                 dp[i] = max(dp[i], dp[i-2*prime[j]] + prime[j]);
30         }
31     }
32     cout << dp[n];
33     return 0;
34 }
```

## ADV-206. 不大的数

### 问题描述

在当今的大数据时代，超大数的高精度计算已经成为众多领域的热门研究之一。现在T校也想在此领域有所造诣已造福于全社会，然而由于时间有限，所以短时间内难以找出大数计算的通用算法，于是学校找到了同学中的“神霸”——你来帮忙，并仅要求你能在数并不算大的时候给出结果。又出于某种特殊需要，也并不要求你给出数的全部结果，而只是要求结果的前10位（注意不是后10位），并考虑到

2的幂次的特殊性和典型性，所以要计算的数均为2的幂次。

输入格式

一个自然数n。

输出格式

2的n次幂的前10位。

样例1 输入

60

样例1 输出

1152921504

样例2 输入

60000

样例2 输出

6305794870

数据规模和约定

$0 \leq n \leq 10000000$

注释

=。 =

分析：本来想用位运算。。后来发现做不出来。。然后换了种方法做，AC了~

1.当乘以2的次数超过或者等于34次的时候，就会超过10位数字。用一个double型变量保存要求的数，然后再最后转为long long int 型。(至于为什么要用double型，因为第二步的除以1000相当于乘以1024，为了避免失去精度使得除法运算后的结果被转为int后不精准~~)为了防止超过10位数字，这个时候可以采取除法的方式让它位数控制在double的不溢出范围之内~~

2.已知2的10次方是1024，也就是说对于每一个是10的倍数i，就相当于给要求的数增加了1024倍，这个时候可以采取除以1000的方式保证它不溢出~

3.但是因为数据规模太大了，有一千万大小，所以每一个 $1024/1000=1.024$ ，当乘以1.024的个数超过97次方的时候( $1.024^{97} = 9.979201547674$ )，就会等于让要求的数乘以了10.为了保证不溢出，可以在乘以了每970个2的时候，让要求的数除以10保证不会溢出~（如果想要更精准的可以用计算器得到： $1.024^{97.1} = 10.00289683498$ ）就是每乘以971个2后除以10~

4.此时用上述办法得到的数如果在数据规模大的时候可能会有10位或者11位，这时候可以统计一下这个double型数字的位数，然后进行除以10的运算把它变为前面只有10位数。

5.此时转换为long long int型输出即可~

```
1  #include <iostream>
2  using namespace std;
3  int main() {
```

```

4      int n;
5      cin >> n;
6      double t = 1.0;
7      for(int i = 1; i <= n; i++) {
8          t = t * 2;
9          if(i >= 34 && i % 10 == 0) {
10             t = t / 1000;
11         }
12         if(i % 971 == 0) {
13             t = t / 10;
14         }
15     }
16     long long int temp = t;
17     int sum = 0;
18     while(temp) {
19         temp = temp / 10;
20         sum++;
21     }
22     while(sum > 10) {
23         t = t / 10;
24         sum--;
25     }
26     long long int d = t;
27     printf("%lld", d);
28     return 0;
29 }

```

## ADV-207. 最长字符序列

### 最长字符序列

#### 问题描述

设 $x(i)$ ,  $y(i)$ ,  $z(i)$ 表示单个字符, 则 $X=\{x(1)x(2).....x(m)\}$ ,  $Y=\{y(1)y(2).....y(n)\}$ ,  $Z=\{z(1)z(2).....z(k)\}$ , 我们称其为字符序列, 其中 $m, n$ 和 $k$ 分别是字符序列 $X$ ,  $Y$ ,  $Z$ 的长度, 括号()中的数字被称作字符序列的下标。

如果存在一个严格递增而且长度大于0的下标序列 $\{i_1, i_2, .....i_k\}$ , 使得对所有的 $j=1, 2, .....k$ , 有 $x(i_j)=z(j)$ , 那么我们称 $Z$ 是 $X$ 的字符子序列。而且, 如果 $Z$ 既是 $X$ 的字符子序列又是 $Y$ 的字符子序列, 那么我们称 $Z$ 为 $X$ 和 $Y$ 的公共字符序列。

在我们今天的问题中, 我们希望计算两个给定字符序列 $X$ 和 $Y$ 的最大长度的公共字符序列, 这里我们只要求输出这个最大长度公共子序列对应的长度值。

举例来说, 字符序列 $X=abcd$ ,  $Y=acde$ , 那么它们的最大长度为3, 相应的公共字符序列为 $acd$ 。

#### 输入格式

输入一行, 用空格隔开的两个字符串

#### 输出格式

输出这两个字符序列对应的最大长度公共字符序列的长度值

#### 样例输入

aAbB aabb

样例输出

2

数据规模和约定

输入字符串长度最长为100，区分大小写。

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      string a, b;
6      cin >> a >> b;
7      int dp[110][100] = {0};
8      for (int i = 1; i <= a.length(); i++) {
9          for (int j = 1; j <= b.length(); j++) {
10             if (a[i - 1] == b[j - 1])
11                 dp[i][j] = dp[i - 1][j - 1] + 1;
12             else dp[i][j] = max(dp[i][j - 1], dp[i - 1][j]);
13         }
14     }
15     cout << dp[a.length()][b.length()];
16     return 0;
17 }
```

## ADV-208. 矩阵相乘

问题描述

小明最近在为线性代数而头疼，线性代数确实很抽象（也很无聊），可惜他的老师正在讲这矩阵乘法这一段内容。

当然，小明上课打瞌睡也没问题，但线性代数的习题可是很可怕的。

小明希望你来帮他完成这个任务。

现在给你一个 $a_i$ 行 $a_j$ 列的矩阵和一个 $b_i$ 行 $b_j$ 列的矩阵，

要你求出他们相乘的积（当然也是矩阵）。

(输入数据保证 $a_j=b_i$ ,不需要判断)

输入格式

输入文件共有 $a_i+b_i+2$ 行，并且输入的所有数为整数（long long范围内）。

第1行：  $a_i$  和  $a_j$

第2~ $a_i+2$ 行： 矩阵a的所有元素

第 $a_i+3$ 行：  $b_i$  和  $b_j$

第 $a_i+3 \sim a_i+b_i+3$ 行：矩阵b的所有元素

输出格式

输出矩阵a和矩阵b的积（矩阵c）

( $a_i$ 行 $b_j$ 列)

样例输入

2 2

12 23

45 56

2 2

78 89

45 56

样例输出

1971 2356

6030 7141

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int main() {
5      int a, b;
6      cin >> a >> b;
7      vector<vector<long long int> > A(a, vector<long long int>(b));
8      for(int i = 0; i < a; i++)
9          for(int j = 0; j < b; j++)
10             cin >> A[i][j];
11     int c, d;
12     cin >> c >> d;
13     vector<vector<long long int> > B(c, vector<long long int>(d));
14     for(int i = 0; i < c; i++)
15         for(int j = 0; j < d; j++)
16             cin >> B[i][j];
17     vector<vector<long long int> > C(a, vector<long long int>(d));
18     for(int i = 0; i < a; i++) {
19         for(int j = 0; j < d; j++) {
20             C[i][j] = 0;
21             for(int k = 0; k < b; k++)
22                 C[i][j] += A[i][k] * B[k][j];
23             cout << C[i][j] << " ";
24         }
25         cout << endl;
26     }
```

```
27     return 0;
28 }
```

## ADV-209. c++\_ch02\_04

### 问题描述

输出1~100间的质数并显示出来。注意1不是质数。

### 输出格式

每行输出一个质数。

2

3

...

97

```
1  #include <iostream>
2  using namespace std;
3  bool isprime(int n) {
4      if(n <= 1)
5          return false;
6      for(int i = 2; i * i <= n; i++) {
7          if(n % i == 0)
8              return false;
9      }
10     return true;
11 }
12
13 int main() {
14     for(int i = 1; i <= 100; i++) {
15         if(isprime(i))
16             cout << i << endl;
17     }
18     return 0;
19 }
```

## ADV-210. 2-1屏幕打印

### 样例输出

\*\*\*\*\*

\* My first C program \*

\*\*\*\*\*

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      cout << "*****\n* My first C program
   *\n*****";
5      return 0;
6  }

```

## ADV-211. 2-2整数求和

基于例子3，写一个程序，实现整数求和：

样例输入

3 4

样例输出

7

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b;
5      cin >> a >> b;
6      cout << a + b;
7      return 0;
8  }

```

## ADV-212. 3-1课后习题2

问题描述

编写一个程序，接受用户输入的10个整数，输出它们的和。

输出格式

要求用户的输出满足的格式。

例：输出1行，包含一个整数，表示所有元素的和。

样例输入

1 2 3 4 5 6 7 8 9 10

样例输出

55

数据规模和约定

输入数据中每一个数的范围。

例：输入数<100000。



```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int sum = 0;
5      for(int i = 0; i < 10; i++) {
6          int t;
7          cin >> t;
8          sum += t;
9      }
10     cout << sum;
11     return 0;
12 }

```

## ADV-214. 3-3求圆面积表面积体积

### 问题描述

接受用户输入的数值，输出以该值为半径的(1)圆面积，(2)球体表面积，(3)球体体积。pi 取值 3.1415926536，结果保留10位小数，每一列占20个字符，左对齐。

### 样例输入

一个满足题目要求的输入范例。

例：

1

### 样例输出

与上面的样例输入对应的输出。

例：(第一行1是输入，第二行是输出)

### 数据规模和约定

所有结果在double类型的表示范围内。

```

1  #include <iostream>
2  #define pi 3.1415926536
3  using namespace std;
4  int main() {
5      double a, b, c;
6      double r;
7      cin >> r;
8      a = pi * r * r;
9      b = 4 * pi * r * r;
10     c = 4.0 / 3 * pi * r * r * r;
11     printf("%-20.10lf%-20.10lf%-20.10lf", a, b, c);
12     return 0;
13 }

```

## ADV-221. 7-1用宏求球的体积

### 问题描述

使用宏实现计算球体体积的功能。用户输入半径，系统输出体积。不能使用函数， $\pi=3.1415926$ ,结果精确到小数点后五位。

### 样例输入

一个满足题目要求的输入范例。

例：

1.0

### 样例输出

与上面的样例输入对应的输出。

例：

```
1.0
4.18879请按任意键继续. . .
```

### 数据规模和约定

输入数据中每一个数的范围。

数据表示采用double类型。

```
1  #include <iostream>
2  using namespace std;
3  #define PI 3.1415926
4  int main() {
5      double r;
6      cin >> r;
7      printf("%.5f", 4 * r * r * r * PI / 3);
8      return 0;
9  }
```

## ADV-222. 7-2求arccos值

### 问题描述

利用标准库中的 $\cos(x)$ 和 $\fabs(x)$ 函数实现 $\arccos(x)$ 函数， $x$ 取值范围是 $[-1, 1]$ ，返回值为 $[0, \pi]$ 。要求结果准确到小数点后5位。(PI = 3.1415926)

提示：要达到这种程度的精度需要使用double类型。

### 样例输入

0.5

### 样例输出

```
0.5
1.04720请按任意键继续. . .
```

数据规模和约定

$-1 \leq x \leq 1, 0 \leq \arccos(x) \leq \pi$ 。

分析：二分逼近，求答案，精确到小数点后6位即可~

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  #define PI 3.1415926
5  #define min 0.000001
6  double x;
7  double arccos(double l, double r) {
8      if (r - l < min) return l;
9      double mid = (l + r) / 2;
10     if (cos(mid) == x) return mid;
11     else if (cos(mid) > x) return arccos(mid, r);
12     else return arccos(l, mid);
13 }
14 int main() {
15     cin >> x;
16     printf("%.5f", arccos(0, PI));
17     return 0;
18 }
```

## ADV-223. 8-1因式分解

问题描述

设计算法，用户输入合数，程序输出若个素数的乘积。例如，输入6，输出2\*3。输入20，输出2\*2\*5。

样例

与上面的样例输入对应的输出。

例：

```
120
2*2*2*3*5请按任意键继续. . .
```

```
13
13请按任意键继续. . .
```

数据规模和约定

输入数据中每一个数在int表示范围内。

分析：1.先筛出50000以内的素数，用n依次对每个素数取余，获得自身的所有因子  
2.最后，如果n==1，说明还剩下一个比50000大的素数因子~

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  int main() {
5      int n, p[50001] = {0};
6      cin >> n;
7      vector<int> vp, ans;
8      for (int i = 2; i <= 50000; i++) {
9          if (p[i] == 0) {
10             vp.push_back(i);
11             for (int j = 1; j * i <= 50000; j++) {
12                 p[i * j] = -1;
13             }
14         }
15     }
16     for (int i = 0; i < vp.size(); i++) {
17         int t = vp[i];
18         while (n % t == 0) {
19             ans.push_back(t);
20             n /= t;
21         }
22     }
23     if (n != 1) ans.push_back(n);
24     for (int i = 0; i < ans.size(); i++) {
25         if (i != 0) cout << ' ';
26         cout << ans[i];
27     }
28     return 0;
29 }

```

## ADV-224. 9-1九宫格

### 问题描述

九宫格。输入1-9这9个数字的一种任意排序，构成3\*3二维数组。如果每行、每列以及对角线之和都相等，打印1。否则打印0。

### 样例输出

与上面的样例输入对应的输出。

例：

```

4 9 2
3 5 7
8 1 6
1请按任意键继续. . .

```

### 数据规模和约定

输入1-9这9个数字的一种任意排序。

```

1  #include <iostream>
2  #include <algorithm>
3  #include <vector>
4  #include <queue>
5  #include <cmath>
6  using namespace std;
7  int main() {
8      int a[10], ans = 1;
9      for (int i = 1; i <= 9; i++)
10         cin >> a[i];
11     if (a[1] + a[2] + a[3] != 15) ans = 0;
12     if (a[4] + a[5] + a[6] != 15) ans = 0;
13     if (a[7] + a[8] + a[9] != 15) ans = 0;
14     if (a[1] + a[4] + a[7] != 15) ans = 0;
15     if (a[2] + a[5] + a[8] != 15) ans = 0;
16     if (a[3] + a[6] + a[9] != 15) ans = 0;
17     if (a[1] + a[5] + a[9] != 15) ans = 0;
18     if (a[7] + a[5] + a[3] != 15) ans = 0;
19     cout << ans;
20     return 0;
21 }

```

## ADV-225. 9-2 文本加密

### 问题描述

先编写函数EncryptChar,按照下述规则将给定的字符c转化（加密）为新的字符：“A”转化“B”，“B”转化为“C”，... ..“Z”转化为“a”，“a”转化为“b”，... ..，“z”转化为“A”，其它字符不加密。编写程序，加密给定字符串。

### 样例输出

与上面的样例输入对应的输出。

例：

```

helloWorld!
ifmmpXpsme!
请按任意键继续. . .

```

### 数据规模和约定

输入数据中每一个数的范围。

例：50个字符以内无空格字符串。

```

1  n#include <iostream>
2  #include <algorithm>
3  #include <vector>
4  #include <cctype>
5  #include <map>
6  using namespace std;

```

```

7  string encrypt(string s) {
8      for (int i = 0; i < s.length(); i++) {
9          if (s[i] == 'z') s[i] = 'a';
10         else if (s[i] == 'Z') s[i] = 'A';
11         else if (isalpha(s[i])) s[i]++;
12     }
13     return s;
14 }
15 int main() {
16     string s;
17     cin >> s;
18     cout << encrypt(s);
19     return 0;
20 }

```

## ADV-226. 9-3摩尔斯电码

### 问题描述

摩尔斯电码破译。类似于乔林教材第213页的例6.5，要求输入摩尔斯码，返回英文。请不要使用“zlib.h”，只能使用标准库函数。用‘\*’表示‘.’，中间空格用‘|’表示，只转化字符表。

摩尔斯码定义见：<http://baike.baidu.com/view/84585.htm?fromId=253988>。

### 提示

清橙进行评测时，输入是以EOF结尾的，而不是换行符。（EOF不是一个字符，“以EOF结尾”是一种通俗但不严谨的说法。）因此可以通过以下方式之一获取输入：

1. 一次读入整行字符串，再进行后续解析。
2. 使用getchar或scanf一次读入一个字符，通过它们的返回值判断输入结束。

### 字母

| 字符 | 电码符号    | 字符 | 电码符号    | 字符 | 电码符号    | 字符 | 电码符号    |
|----|---------|----|---------|----|---------|----|---------|
| A  | . -     | B  | - . . . | C  | - . - . | D  | - . .   |
| E  | .       | F  | . . - . | G  | - - .   | H  | . . . . |
| I  | . .     | J  | . - - - | K  | - . -   | L  | . - . . |
| M  | - -     | N  | - .     | O  | - - -   | P  | . - - . |
| Q  | - - . - | R  | . - .   | S  | . . .   | T  | -       |
| U  | . . -   | V  | . . . - | W  | . - -   | X  | - . . - |
| Y  | - . - - | Z  | - - . . |    |         |    |         |

### 样例输出

```

****|*|*-**|*-**|---
hello请按任意键继续. . .

```

```

1  #include <iostream>
2  #include <vector>
3  #include <map>
4  using namespace std;
5  int main() {
6      map<string, char> m;
7      m["*-"] = 'a';
8      m["-***"] = 'b';
9      m["-*-*"] = 'c';
10     m["-***"] = 'd';
11     m["*"] = 'e';
12     m["**-*"] = 'f';
13     m["--*"] = 'g';
14     m["*****"] = 'h';
15     m["**"] = 'i';
16     m["*---"] = 'j';
17     m["-*-"] = 'k';
18     m["*-***"] = 'l';
19     m["--"] = 'm';
20     m["-*"] = 'n';
21     m["---"] = 'o';
22     m["*---*"] = 'p';
23     m["--*-"] = 'q';
24     m["*-*-"] = 'r';
25     m["***"] = 's';
26     m["-"] = 't';
27     m["***-"] = 'u';
28     m["****-"] = 'v';
29     m["*---"] = 'w';
30     m["-***-"] = 'x';
31     m["-*---"] = 'y';
32     m["--***"] = 'z';
33     string s;
34     vector<string> v;
35     cin >> s;
36     s += '|';
37     for(int i = 0; i < s.length(); i++){
38         string t;
39         for(; s[i] != '|'; i++)
40             t += s[i];
41         v.push_back(t);
42     }
43     for(int i = 0; i < v.size(); i++)
44         cout << m[v[i]];
45     return 0;
46 }

```

## ADV-227. 11-1实现strcmp函数

## 问题描述

自己实现一个比较字符串大小的函数，也即实现strcmp函数。函数：int myStrcmp(char \*s1,char \*s2) 按照ASCII顺序比较字符串s1与s2。若s1与s2相等返回0，s1>s2返回1，s1<s2返回-1。具体来说，两个字符串自左向右逐个字符相比（按ASCII值大小相比较），直到出现不同的字符或遇'\0'为止（注意'\0'值为0，小于任意ASCII字符）。如：

"A"<"B"

"a">"A"

"computer">"compare"

"hello"<"helloworld"

## 样例输出

```
hello
helloworld
-1
请按任意键继续. . .
```

## 数据规模和约定

字符串长度<100。

```
1  #include <iostream>
2  #include <algorithm>
3  #include <vector>
4  #include <cctype>
5  #include <map>
6  using namespace std;
7  int f(string s1, string s2) {
8      s1 += char(0);
9      s2 += char(0);
10     for (int i = 0; i < s1.length() && i < s2.length(); i++) {
11         if (s1[i] < s2[i]) return -1;
12         if (s1[i] > s2[i]) return 1;
13     }
14     return 0;
15 }
16 int main() {
17     string s1, s2;
18     cin >> s1 >> s2;
19     cout << f(s1, s2);
20     return 0;
21 }
```

## ADV-228. 11-2删除重复元素

### 问题描述

为库设计新函数DelPack，删除输入字符串中所有的重复元素。不连续的重复元素也要删除。要求写成函数，函数内部使用指针操作。



样例输入

1223445667889

样例输出

13579

样例输入

else

样例输出

ls

数据规模和约定

字符串数组最大长度为100。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a[200] = {0};
5      string s;
6      cin >> s;
7      for (int i = 0; i < s.length(); i++)
8          a[s[i]]++;
9      for (int i = 0; i < s.length(); i++)
10         if (a[s[i]] == 1) cout << s[i];
11     return 0;
12 }
```

## ADV-233. 队列操作

问题描述

队列操作题。根据输入的操作命令，操作队列（1）入队、（2）出队并输出、（3）计算队中元素个数并输出。

输入格式

第一行一个数字N。

下面N行，每行第一个数字为操作命令（1）入队、（2）出队并输出、（3）计算队中元素个数并输出。

输出格式

若干行每行显示一个2或3命令的输出结果。注意：2.出队命令可能会出现空队出队（下溢），请输出“no”，并退出。

样例输入

7  
1 19  
1 56  
2  
3  
2  
3  
2

样例输出

19  
1  
56  
0  
no

数据规模和约定

$1 \leq N \leq 50$

分析：用C++的STL队列实现～

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4  int main() {
5      int n;
6      cin >> n;
7      queue<int> q;
8      for (int i = 0; i < n; i++) {
9          int query, temp;
10         cin >> query;
11         if (query == 1) {
12             cin >> temp;
13             q.push(temp);
14         } else if (query == 2) {
15             if (q.empty()) {
16                 cout << "no" << endl;
17                 return 0;
18             } else {
19                 cout << q.front() << endl;
20                 q.pop();
21             }
22         } else if (query == 3) {
23             cout << q.size() << endl;
24         }
25     }
26     return 0;
27 }
```

## ADV-234. 字符串跳步

### 问题描述

给定一个字符串，你需要从第start位开始每隔step位输出字符串对应位置上的字符。

### 输入格式

第一行一个只包含小写字母的字符串。

第二行两个非负整数start和step，意义见上。

### 输出格式

一行，表示对应输出。

### 样例输入

abcdefg

2 2

### 样例输出

ceg

### 数据规模和约定

start从0开始计数。

字符串长度不超过100000。

### 提示

读入上有问题，可以参照字符串进位。

尝试不要出现以下代码：for (int i = 0; i < (int) S.size(); ++i)

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string s;
6      int star, step;
7      cin >> s >> star >> step;
8      for(int i = 0; star+ i * step <= s.size(); i++)
9          cout << s[star + i * step];
10     return 0;
11 }
```

## ADV-235. 阶乘差

### 问题描述

给定n和m以及p，保证 $n \geq m$ ，求 $(n! - m!)$ 对p取余的结果。

### 输入格式

一行三个正整数n,m,p。

输出格式

一行一个非负整数表示结果。

样例输入

3 2 10

样例输出

4

数据规模和约定

$n, m \leq 20$ ,  $p \leq 10000$ .

分析：先取余再相减，减完为了避免出现负数的情况，加上p再取余~

```
1  #include <iostream>
2  using namespace std;
3  int n, m, p;
4  int f(int n){
5      int ans = 1;
6      for(int i = 1; i <= n; i++)
7          ans = ans * i % p;
8      return ans;
9  }
10 int main() {
11     cin >> n >> m >> p;
12     cout << (f(n)-f(m) + p) % p;
13     return 0;
14 }
```

## ADV-237. 三进制数位和

问题描述

给定L和R，你需要对于每一个6位三进制数（允许前导零），计算其每一个数位上的数字和，设其在十进制下为S。

一个三进制数被判断为合法，当且仅当S为质数，或者S属于区间[L,R]。

你的任务是给出合法三进制数的个数

输入格式

一行两个非负整数L,R。

输出格式

一行一个非负整数表示答案。

样例输入

0 0

样例输出

数据规模和约定

保证 $0 \leq L < R \leq 12$ 。

提示

判断x是否为质数核心代码: `for (int i = 2; i * i <= x; ++i) if (x % i == 0) { /*你猜? */ }`

分析: 所有6为三进制数一共有 $3^6 - 1 = 729$ 个, 把这729个数转成三进制再判断~

```

1  #include <iostream>
2  using namespace std;
3  bool isprime(int n) {
4      if(n == 1 || n == 0) return false;
5      for (int i = 2; i * i <= n; i++)
6          if (n % i == 0) return false;
7      return true;
8  }
9  int main() {
10     int l, r, ans = 0;
11     cin >> l >> r;
12     for (int i = 0; i < 729; i++) {
13         int sum = 0, j = i;
14         while(j) {
15             sum += j % 3;
16             j /= 3;
17         }
18         if(isprime(sum) || (l <= sum && sum <= r))
19             ans++;
20     }
21     cout << ans << endl;
22     return 0;
23 }
```

## ADV-238. P0101

一个水分子的质量是 $3.0 \times 10^{-23}$ 克, 一夸脱水的质量是950克。写一个程序输入水的夸脱数n ( $0 \leq n \leq 1e10$ ), 然后输出水分子的总数。

输入

109.43

输出

3.465283E+027

分析: 1.结果为0的时候特判

2.结果在0~1之间的, 获得小数点后面的0的个数

3.结果大于1的, 要获得小数点前面多的整数位的位数

4.注意位数可能超过int, 要用long long存

5.获得位数是可以用取以10为底对数的方式代替循环~

```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5      double n, ans;
6      cin >> n;
7      ans = n * 950 / 3.0;
8      if(ans == 0) {
9          printf("0.000000E+000");
10     } else if (ans >= 1){
11         int num = log10(ans);
12         printf("%6fE+%03lld", ans/pow(10,num), 23+num);
13     }else{
14         int num = log10(ans)*-1 + 1;
15         printf("%6fE+%03lld", ans*pow(10,num), 23-num);
16     }
17     return 0;
18 }

```

## ADV-239. P0102

用户输入三个字符，每个字符取值范围是0-9, A-F。然后程序会把这三个字符转化为相应的十六进制整数，并分别以十六进制，十进制，八进制输出，十六进制表示成3位，八进制表示成4位，若不够前面补0。（不考虑输入不合法的情况）

输入

1D5

输出

（注意冒号后面有一个空格）

Hex: 0x1D5

Decimal: 469

Octal: 0725

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int f(char c) {
5      if ('0' <= c && c <= '9') return c - '0';
6      else return c - 'A' + 10;
7  }
8  int main() {
9      string s, anso;
10     int ansd, t;
11     cin >> s;
12     t = ansd = f(s[0]) * 16 * 16 + f(s[1]) * 16 + f(s[2]);
13     while (t) {
14         anso = (char) (t % 8 + '0') + anso;
15         t /= 8;
16     }

```

```

17     string s0(4-anso.length(),'0');
18     printf("Hex: 0x%s\n", s.c_str());
19     printf("Decimal: %d\n", ansd);
20     printf("Octal: %s%s\n",s0.c_str(), anso.c_str());
21     return 0;
22 }

```

## PREV-5. 错误票据

### 问题描述

某涉密单位下发了某种票据，并要在年终全部收回。

每张票据有唯一的ID号。全年所有票据的ID号是连续的，但ID的开始数码是随机选定的。

因为工作人员疏忽，在录入ID号的时候发生了一处错误，造成了某个ID断号，另外一个ID重号。

你的任务是通过编程，找出断号的ID和重号的ID。

假设断号不可能发生在最大和最小号。

### 输入格式

要求程序首先输入一个整数N(N<100)表示后面数据行数。

接着读入N行数据。

每行数据长度不等，是用空格分开的若干个（不大于100个）正整数（不大于100000），请注意行内和行末可能有多余的空格，你的程序需要能处理这些空格。

每个整数代表一个ID号。

### 输出格式

要求程序输出1行，含两个整数m n，用空格分隔。

其中，m表示断号ID，n表示重号ID

### 样例输入1

```

2
5 6 8 11 9
10 12 9

```

### 样例输出1

```

7 9

```

### 样例输入2

```

6
164 178 108 109 180 155 141 159 104 182 179 118 137 184 115 124 125 129 168 196
172 189 127 107 112 192 103 131 133 169 158
128 102 110 148 139 157 140 195 197

```

185 152 135 106 123 173 122 136 174 191 145 116 151 143 175 120 161 134 162 190  
149 138 142 146 199 126 165 156 153 193 144 166 170 121 171 132 101 194 187 188  
113 130 176 154 177 120 117 150 114 183 186 181 100 163 160 167 147 198 111 119

样例输出2

105 120

分析：把它们一个个放到集合里面，要是集合的容量不变说明是重复的数字~然后遍历集合，如果集合有一个数字不连续那就是这个数字是错误的~

用了while(cin >> a)后，发现第一个变量给不给都一样。。反正没什么用。。

```
1  #include <iostream>
2  #include <set>
3  using namespace std;
4  int main() {
5      set<int> s;
6      int row, a, m = 0, n = 0, size;
7      cin >> row;
8      while(cin >> a) {
9          size = s.size();
10         s.insert(a);
11         if(s.size() == size) {
12             n = a;
13         }
14     }
15     set<int>::iterator it = s.begin();
16     set<int>::iterator itt = it;
17     for(it++; it != s.end(); it++, itt++) {
18
19         if(*it != *itt + 1) {
20             m = *itt + 1;
21         }
22     }
23     cout << m << " " << n;
24     return 0;
25 }
```

## PREV-6. 翻硬币

问题描述

小明正在玩一个“翻硬币”的游戏。

桌上放着排成一排的若干硬币。我们用 \* 表示正面，用 o 表示反面（是小写字母，不是零）。

比如，可能情形是：\*\*oo\*\*\*oooo

如果同时翻转左边的两个硬币，则变为：oooo\*\*\*oooo

现在小明的问题是：如果已知了初始状态和要达到的目标状态，每次只能同时翻转相邻的两个硬币,那么对特定的局面，最少要翻动多少次呢？



我们约定：把翻动相邻的两个硬币叫做一步操作，那么要求：

输入格式

两行等长的字符串，分别表示初始状态和要达到的目标状态。每行的长度<1000

输出格式

一个整数，表示最小操作步数。

样例输入1

```
*****  
0****0****
```

样例输出1

5

样例输入2

```
*0**0***0***  
*0***0**0***
```

样例输出2

1

分析：贪心，从前往后，如果发现当前硬币和目标硬币不一样，则同时翻转这枚硬币和下一枚硬币～

```
1  #include <iostream>  
2  #include <string>  
3  using namespace std;  
4  int main() {  
5      int cnt = 0;  
6      string s1, s2;  
7      cin >> s1 >> s2;  
8      for (int i = 0; i < s1.length(); i++) {  
9          if (s1[i] != s2[i]) {  
10             if (s1[i] == 'o') s1[i] = '*';  
11             else s1[i] = 'o';  
12             if (s1[i + 1] == 'o') s1[i + 1] = '*';  
13             else s1[i + 1] = 'o';  
14             cnt++;  
15         }  
16     }  
17     cout << cnt;  
18     return 0;  
19 }
```

## PREV-8. 买不到的数目

问题描述

小明开了一家糖果店。他别出心裁：把水果糖包成4颗一包和7颗一包的两种。糖果不能拆包卖。小朋友来买糖的时候，他就用这两种包装来组合。当然有些糖果数目是无法组合出来的，比如要买 10 颗糖。

你可以用计算机测试一下，在这种包装情况下，最大不能买到的数量是17。大于17的任何数字都可以用4和7组合出来。

本题的要求就是在已知两个包装的数量时，求最大不能组合出的数字。

输入格式

两个正整数，表示每种包装中糖的颗数(都不多于1000)

输出格式

一个正整数，表示最大不能买到的糖数

样例输入1

4 7

样例输出1

17

样例输入2

3 5

样例输出2

7

分析：完全背包，且weight==value dp[i]为空间为i时能装的最大value, 从后往前找，如果dp[i]<i则找到该数字~

```
1  #include <iostream>
2  #include <cmath>
3  #include <vector>
4  using namespace std;
5  int main() {
6      int t, n = 0, dp[100001] = {0}, maxn = 100000;
7      vector<int> w(1), v(1);
8      while (cin >> t) {
9          w.push_back(t);
10         v.push_back(t);
11         n++;
12     }
13     for (int i = 1; i <= n; i++) {
14         for (int j = w[i]; j <= maxn; j++) {
15             dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
16         }
17     }
18     dp[0] = -1;
19     for (int i = maxn; i >= 0; i--) {
20         if (dp[i] < i) {
```

```

21         cout << i;
22         return 0;
23     }
24 }
25 return 0;
26 }

```

## PREV-27. 蚂蚁感冒

### 问题描述

长100厘米的细长直杆子上有n只蚂蚁。它们的头有的朝左，有的朝右。

每只蚂蚁都只能沿着杆子向前爬，速度是1厘米/秒。

当两只蚂蚁碰面时，它们会同时掉头往相反的方向爬行。

这些蚂蚁中，有1只蚂蚁感冒了。并且在和其它蚂蚁碰面时，会把感冒传染给碰到的蚂蚁。

请你计算，当所有蚂蚁都爬离杆子时，有多少只蚂蚁患上了感冒。

### 输入格式

第一行输入一个整数n ( $1 < n < 50$ ), 表示蚂蚁的总数。

接着的一行是n个用空格分开的整数  $X_i$  ( $-100 < X_i < 100$ ),  $X_i$ 的绝对值，表示蚂蚁离开杆子左边端点的距离。正值表示头朝右，负值表示头朝左，数据中不会出现0值，也不会出现两只蚂蚁占用同一位置。其中，第一个数据代表的蚂蚁感冒了。

### 输出格式

要求输出1个整数，表示最后感冒蚂蚁的数目。

### 样例输入

```

3
5 -2 8

```

### 样例输出

```

1

```

### 样例输入

```

5
-10 8 -20 12 25

```

### 样例输出

```

3

```

分析：1.两只蚂蚁相遇，他们都掉头，不妨把他们看作交换身体，和各走各的效果一样的

2.记录感冒蚂蚁中，lr在左面往右爬，ll在左面往左爬，rl在右面往左爬，rr在右面往右爬的数量

3.如果感冒蚂蚁往左爬，那lr会被该蚂蚁传染，lr会传染rl。所以，如果lr>0，会传给rl蚂蚁，答案为lr + rl,如果lr= 0那lr不会被传染,更不会传染给rl，答案为1,即最开始的一只感冒的。感冒蚂蚁往右爬同理~

```

1 #include <iostream>
2 #include <cmath>

```

```

3  using namespace std;
4  int main() {
5      int n, cold, t, ll = 0, lr = 0, rl = 0, rr = 0, ans = 0;
6      cin >> n >> cold;
7      for (int i = 0; i < n - 1; i++) {
8          cin >> t;
9          if (t > 0) {
10             if (abs(t) < abs(cold)) lr++;
11             else rr++;
12         } else {
13             if (abs(t) < abs(cold)) ll++;
14             else rl++;
15         }
16     }
17     if (cold < 0) {
18         if (lr > 0) ans = lr + rl + 1;
19         else ans = 1;
20
21     } else {
22         if (rl > 0) ans = rl + lr + 1;
23         else ans = 1;
24
25     }
26     cout << ans;
27     return 0;
28 }

```

## PREV-32. 分糖果

### 问题描述

有n个小朋友围坐成一圈。老师给每个小朋友随机发偶数个糖果，然后进行下面的游戏：  
 每个小朋友都把自己的糖果分一半给左手边的孩子。  
 一轮分糖后，拥有奇数颗糖的孩子由老师补给1个糖果，从而变成偶数。  
 反复进行这个游戏，直到所有小朋友的糖果数都相同为止。  
 你的任务是预测在已知的初始糖果情形下，老师一共需要补发多少个糖果。

### 输入格式

程序首先读入一个整数N( $2 < N < 100$ )，表示小朋友的人数。  
 接着是一行用空格分开的N个偶数（每个偶数不大于1000，不小于2）

### 输出格式

要求程序输出一个整数，表示老师需要补发的糖果数。

### 样例输入

```

3
2 2 4

```

### 样例输出

分析：模拟即可，注意必须等小朋友先分一次糖后才判断，用do while结构

```

1  #include <iostream>
2  using namespace std;
3  int a[150], ahalf[150], n, cnt = 0;
4  bool check() {
5      bool flag = true;
6      for (int i = 1; i < n; i++)
7          if (a[i] != a[i + 1]) flag = false;
8      for (int i = 1; i <= n; i++) {
9          if (a[i] % 2 == 1) {
10             a[i]++;
11             cnt++;
12         }
13     }
14     return flag;
15 }
16 int main() {
17     cin >> n;
18     for (int i = 1; i <= n; i++)
19         cin >> a[i];
20     do {
21         for (int i = 1; i <= n; i++)
22             ahalf[i] = a[i] / 2;
23         for (int i = 1; i < n; i++)
24             a[i] = ahalf[i] + ahalf[i + 1];
25         a[n] = ahalf[n] + ahalf[1];
26     } while (!check());
27     cout << cnt;
28     return 0;
29 }

```

## PREV-37. 分巧克力

### 问题描述

儿童节那天有K位小朋友到小明家做客。小明拿出了珍藏的巧克力招待小朋友们。

小明一共有N块巧克力，其中第i块是 $H_i \times W_i$ 的方格组成的长方形。

为了公平起见，小明需要从这 N 块巧克力中切出K块巧克力分给小朋友们。切出的巧克力需要满足：

1. 形状是正方形，边长是整数
2. 大小相同

例如一块 $6 \times 5$ 的巧克力可以切出6块 $2 \times 2$ 的巧克力或者2块 $3 \times 3$ 的巧克力。

当然小朋友们都希望得到的巧克力尽可能大，你能帮小Hi计算出最大的边长是多少么？

### 输入格式

第一行包含两个整数N和K。(1 ≤ N, K ≤ 100000)  
以下N行每行包含两个整数Hi和Wi。(1 ≤ Hi, Wi ≤ 100000)  
输入保证每位小朋友至少能获得一块1×1的巧克力。

输出格式

输出切出的正方形巧克力最大可能的边长。

样例输入

```
2 10
6 5
5 6
```

样例输出

```
2
```

数据规模和约定

峰值内存消耗（含虚拟机） < 256M

CPU消耗 < 1000ms

请严格按照要求输出，不要画蛇添足地打印类似：“请您输入...”的多余内容。

注意：

main函数需要返回0;

只使用ANSI C/ANSI C++ 标准;

不要调用依赖于编译环境或操作系统的特殊函数。

所有依赖的函数必须明确地在源文件中 #include <xxx>

不能通过工程设置而省略常用头文件。

提交程序时，注意选择所期望的语言类型和编译器类型。

分析：枚举巧克力边长，暴力搜索会超出时，用二分搜索～

```
1  #include <iostream>
2  #include <vector>
3  #include <utility>
4  using namespace std;
5  vector<pair<int, int> > v;
6  int n, k, l = 1, r = 99999999;
7  int check(int a) {
8      int res = 0;
9      for (int j = 0; j < n; j++) {
10         res += (v[j].first / a) * (v[j].second / a);
11         if (res >= k) break;
12     }
13     return res - k;
14 }
15 int main() {
16     scanf("%d%d", &n, &k);
17     v.resize(n);
```

```

18     for (int i = 0; i < n; i++)
19         scanf("%d%d", &v[i].first, &v[i].second);
20     while (l <= r) {
21         int mid = (r + l) / 2;
22         if (check(mid) >= 0 && check(mid + 1) < 0) {
23             cout << mid << endl;
24             return 0;
25         } else if (check(mid) < 0) {
26             r = mid - 1;
27         } else if (check(mid) >= 0) {
28             l = mid + 1;
29         }
30     }
31     return 0;
32 }

```

## PREV-54. 合根植物

### 问题描述

w星球的一个种植园，被分成  $m * n$  个小格子（东西方向m行，南北方向n列）。每个格子里种了一株合根植物。

这种植物有个特点，它的根可能会沿着南北或东西方向伸展，从而与另一个格子的植物合成为一体。如果我们告诉你哪些小格子间出现了连根现象，你能说出这个园中一共有多少株合根植物吗？

### 输入格式

第一行，两个整数m, n，用空格分开，表示格子的行数、列数（ $1 < m, n < 1000$ ）。

接下来一行，一个整数k，表示下面还有k行数据（ $0 < k < 100000$ ）

接下来k行，第行两个整数a, b，表示编号为a的小格子和编号为b的小格子合根了。

格子的编号一行一行，从上到下，从左到右编号。

比如：5 \* 4 的小格子，编号：

```

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20

```

### 样例输入

```

5 4
16
2 3
1 5
5 9
4 8
7 8
9 10
10 11
11 12

```

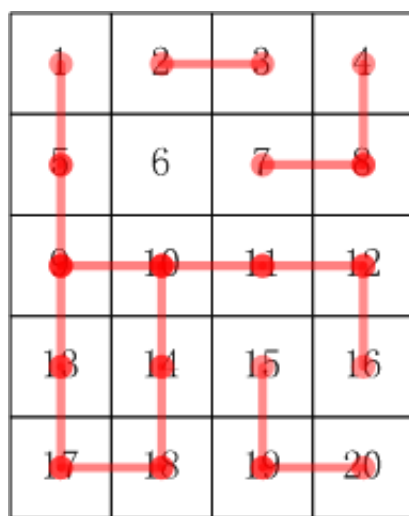
10 14  
12 16  
14 18  
17 18  
15 19  
19 20  
9 13  
13 17

样例输出

5

样例说明

其合根情况参考下图



分析：一个没有套路的并查集～ 求有多少个集合～

```
1  #include <iostream>
2  int a[1000001], ans = 0;
3  using namespace std;
4  int find(int x) {
5      while (x != a[x]) {
6          x = a[x];
7      }
8      int t = x;
9      while (t != a[t]) {
10         int z = t;
11         t = a[t];
12         a[z] = x;
13     }
14     return x;
15 }
16 void uni(int x, int y) {
17     a[find(x)] = find(y);
18 }
```



```
19  int main() {
20      int m, n, k;
21      scanf("%d%d%d", &m, &n, &k);
22      for (int i = 0; i <= m * n; i++)
23          a[i] = i;
24      for (int i = 0; i < k; i++) {
25          int t1, t2;
26          scanf("%d%d", &t1, &t2);
27          uni(t1, t2);
28      }
29      for (int i = 1; i <= m * n; i++)
30          if (find(i) == i) ans++;
31      cout << ans << endl;
32      return 0;
33 }
```